Fused Feature Representation Discovery for High-Dimensional and Sparse Data

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

Abstract

The automatic discovery of a significant low-dimensional feature representation from a given data set is a fundamental problem in machine learning. This paper focuses specifically on the development of the feature representation discovery methods appropriate for high-dimensional and sparse data. We formulate our feature representation discovery problem as a variant of the semi-supervised learning problem, namely, as an optimization problem over unsupervised data whose objective is evaluating the impact of each feature with respect to modeling a target task according to the initial model constructed by using supervised data. The most notable characteristic of our method is that it offers a feasible processing speed even if the numbers of data and features are both in the millions or even billions, and successfully provides a significantly small number of feature sets, *i.e.*, fewer than 10, that can also offer improved performance compared with those obtained with the original feature sets. We demonstrate the effectiveness of our method in experiments consisting of two well-studied natural language processing tasks.

Introduction

The automatic discovery of a significant low-dimensional feature representation from a given data set, which we refer to as '*feature representation discovery*', has been a long-standing goal of machine learning research. Many different feature representation discovery methods have already been developed, and their usefulness has been demonstrated in many areas of real data analysis, including text, speech, image, and signal data processing. For example, PCA, SVD, ICA, and modern variants (Van Der Maaten and Hinton 2008) are typical feature representation discovery methods that seek a low-dimensional representation via feature/data matrix decomposition. One example that directly seeks sparse representation is sparse coding (Zhang et al. 2011). Standard clustering methods have also been utilized to capture reduced representations, and have mainly been applied to tasks with discrete feature spaces (Turian, Ratinov, and Bengio 2010). More recently, several new ideas have been proposed including 'word-codebook' (Kuksa and Qi 2010), which tries to capture an abstract of words as a low-dimensional real valued vector, and deep learning (Hinton 2007), which seeks a representation that captures higher level, abstract features of the given data using a multi-layer neural network.

Let X be an $M \times N$ feature/data matrix, where M is the number of features, and N is the number of data points. In this paper, we focus on a situation where the task at hand is extremely large, *i.e.*, M and N are both in the millions or even billions. Today, we often encounter such large-scale problems, especially in text processing and bio-informatics, since the automated collection of (unsupervised) data is rapidly increasing, and many fine-grained features must be incorporated to improve task performance. Thus, developing specialized algorithms suitable for dealing with large-scale problems is an important research topic in machine learning.

The difficulty presented by large problems mainly originates in the computational cost of the solver algorithms. For example, polynomial time algorithms are obviously infeasible when the numbers of features and/or data points are in the millions. Another difficulty may derive from data sparsity, *i.e.*, more than 90% or even 99% of the elements in matrix X are zero, since, in general, large-scale problems tend to be very sparse problems. Under this condition, for example, PCA, ICA and their variants are essentially useless since most of the data points are orthogonal to each other. Additionally, obtained new feature representations should satisfy the condition that they can be calculated without incurring a large additional computational cost.

Against this background, the goal of this paper is to provide a feasible and appropriate method for tackling extremely large feature representation discovery problems. First, we formulate our feature representation discovery problem as a variant of a semi-supervised learning problem, namely, an optimization problem over unsupervised data whose objective is to evaluate the impact of each feature with respect to modeling a target task according to the initial model, which is constructed by using supervised data. Our method has the following three main characteristics; (1) it has the ability to handle a large number of data *i.e.*, the order of billions, since our method is designed to work in distributed computing environments, (2) it can work appropriately even if the original feature set is an infinite set, and (3) the resultant new feature representation generated by our method consists of only a very small number of features, *i.e.*,

fewer than 10, which are combinatorially generated from original feature sets. Thus, our method is considered to be suitable for large problems.

We evaluate the effectiveness of our method on two well-studied natural language processing tasks, namely, dependency parsing and named entity recognition. Our method provides a feasible processing speed even when there are billions of data points and features. Moreover, using a fused feature set as the feature set for supervised learning significantly improves the task performance compared with using the original (canonical) feature set of the given target task.

Fused Feature Representation Discovery

Suppose we have feature set $\mathcal{F}$, *i.e.*, a canonical feature set widely used in the target task. In this paper, we assume that each feature is represented as a function. For example, $\mathcal{F} = \{f_i(\cdot)\}_{i=1}^{|\mathcal{F}|}$, where $f_i(\cdot)$ represents the i-th feature function, and $|F|$ represents the numbers of features in $\mathcal{F}$. This paper refers to $\mathcal{F}$ as the '**original feature set**'. Then, we define our feature representation discovery problem as follows:

Definition 1 (Feature representation discovery)**.** *Let $\mathcal{H}$ represent a feature set. The problem of feature representation discovery is to find a new (reduced and augmented) feature set $\mathcal{H}$ from the original feature set $\mathcal{F}$. We assume that $\mathcal{H}$ improves the task performance compared with using $\mathcal{F}$ even if $|\mathcal{H}| \leq |\mathcal{F}|$, and generally $|\mathcal{H}| \ll |\mathcal{F}|$ for large $|\mathcal{F}|$.*

Fused Feature Representation To derive our task definition in detail, we first define a feasible value set $\mathcal{S}_K$:

Definition 2 (feasible value set)**.** *Suppose K is a finite positive integer $(0 < K < \infty)$, and v_k for all k are positive real values $(0 < v_k < \infty)$, where $v_{k-1} > v_k$. Then, we define a finite set of values $\mathcal{S}_K$, which we call a feasible value set, as: $\mathcal{S}_K = \bigcup_{k=1}^{K} \{v_k\} \cup \{0\}$.*

For example, if $K = 4$, then $\mathcal{S}_{K=4} = \{v_1, v_2, v_3, v_4, 0\}$, where $v_1 > v_2 > v_3 > v_4 > 0$. To simplify the discussion, let us only consider the case where $f_i(\cdot) \in \mathcal{F}$ for all i are all binary feature functions.

Assumption 3 (binary feature function[1])**.** *$f_i(\cdot) \in \mathcal{F}$ for all i only returns the value 0 or 1.*

Then, we call the new feature set $\mathcal{H}$ proposed in this paper a '**fused feature set**' to distinguish it from other formulations, and define it as follows:

Definition 4 (fused feature set $\mathcal{H}$)**.** *Let us first determine $|\mathcal{H}|$ explained above, and set $|\mathcal{H}| = K$. Then, let $\mathcal{F}_k$ be the k-th subset of $\mathcal{F}$, where $\bigcup_{k=1}^{K+1} \mathcal{F}_k = \mathcal{F}$ and $\bigcap_{k=1}^{K+1} \mathcal{F}_k = \emptyset$ hold. Let $\mathcal{F}'_k = \{\delta_i f_i(\cdot) | f_i(\cdot) \in \mathcal{F}_k \subseteq \mathcal{F}, \delta_i \in \{-1, 1\}\}$. Then, we define the fused feature set $\mathcal{H}$ induced from $\mathcal{F}$ as $\mathcal{H} = \{h_k(\cdot)\}_{k=1}^{K}$, where $h_k(\cdot) = v_k \sum_{\delta f(\cdot) \in \mathcal{F}'_k} \delta f(\cdot)$.*

The form of $\mathcal{H}$ defined by Def. 4 is essentially identical to the several conventional methods such as PCA, namely, the form of the linear combination of the original features. However, note that each $f(\cdot)$ is always assigned to only one $\mathcal{F}_k$. Thus, the problem of constructing $\mathcal{H}$ can be seen as weighted hard clustering with a feature discard operation. $\mathcal{F}_{K+1}$ indicates the set of discarded original features that are not to considered with $\mathcal{H}$. Moreover, $\mathcal{H}$ restricts to have a single coefficient v_k for the k-th fused feature in $\mathcal{H}$. These are the main distinguished properties of fused feature representation from the conventional methods. The possible advantages of using a fused feature set are as follows; (1) it retains a sparse feature representation if the original feature set is sparse. (2) The memory required for keeping the mapping function from the original feature set to the fused feature set is relatively small even if the numbers of data and original features are millions or billions. (3) According to Def. 4, the calculation cost does not increase from that needed when using the original feature set. These properties indicate that the fused feature set is an appropriate representation when the original feature set is very large.

Finally, the goal of this paper, namely to find a better fused feature set $\mathcal{H}$, can be interpreted as follows:

Definition 5 (fused feature representation discovery)**.** *The problem of fused feature representation discovery is defined as finding the three-tuple $(\mathcal{P}, \mathbf{v}, \boldsymbol{\delta})$, where $\mathcal{P} = (\mathcal{F}_k)_{k=1}^{K}$ is the feature partition, $\mathbf{v} = (v_k)_{k=1}^{K}$ is their weights, and $\boldsymbol{\delta} = (\delta_i)_{i=1}^{|\mathcal{F}|}$ is the sign of the corresponding original features.*

Estimation of Impact of Features In accordance with Def. 5, this section defines the $(\mathcal{P}, \mathbf{v}, \boldsymbol{\delta})$-estimation problem for the fused feature representation discovery. To obtain a better $(\mathcal{P}, \mathbf{v}, \boldsymbol{\delta})$, we formulate the $(\mathcal{P}, \mathbf{v}, \boldsymbol{\delta})$-estimation problem as a form of semi-supervised learning, which utilizes both supervised data $\mathcal{D}_L$ and unsupervised data $\mathcal{D}$, assuming that both $\mathcal{D}_L$ and $\mathcal{D}$ of the target task are available, where the quantity of unsupervised data is considered to be very large, whereas that of supervised data is relatively small.

In the first stage of preparation, we build an **initial model** by using $\mathcal{D}_L$ and a typical supervised learning method with a canonical feature set, $\mathcal{F}$. It is worth noting here that the concept of using a supervised model as an initial model for estimating a certain kind of 'feature expectations' over unsupervised data has become a widely used technique in the context of recently developed feature representation discovery and semi-supervised learning methods (Druck and McCallum 2010; Suzuki et al. 2009; Kuksa and Qi 2010). Our method follows their successful strategy.

Another assumption is that the target task can be decomposed into a set of binary decisions. Let Y be the total number of binary decisions in the target task, and y represent an index of a binary decision, that is, $1 \leq y \leq Y$. Then, let $r(\mathbf{x}, y)$ represent the y-th binary decision of a given input $\mathbf{x}$ obtained from the initial model. For example, if the target problem is a Y-class classification problem, then it can be represented as a set of Y-units of binary decisions. In this case, $r(\mathbf{x}, y)$ indicates the decision of the y-th class given input $\mathbf{x}$, where $\sum_y r(\mathbf{x}, y) = 1$. Moreover, a structured prediction problem is defined as the combination of many small sub-problems. We can generally decompose it into a set of many binary decisions. We define $\mathcal{Y}(\mathbf{x})$ as a set

[1] We note that our method can be extended to handle real value features by utilizing the idea of weak hypotheses in the context of boosting algorithms. We convert all the original features into binary weak hypotheses before applying our method.

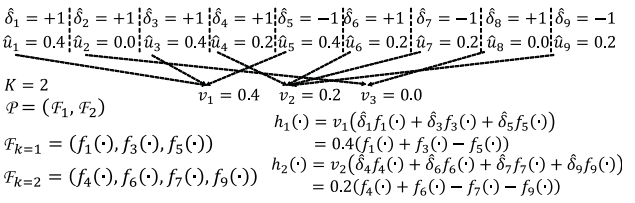


Figure 1: Example of making fused feature set where $K = 2$.

of all binary decisions given input $\mathbf{x}$ since the set of binary decisions is determined independently by each input $\mathbf{x}$ with the structured prediction problem. This paper refers to $\tilde{r}(\cdot)$ as '**reference function**', and $\tilde{r}(\mathbf{x}, y) = 2r(\mathbf{x}, y) - 1$. We define the $(\mathcal{P}, \mathbf{v}, \boldsymbol{\delta})$-estimation problem as a *feature-wise* loss minimization problem that evaluates the sum of the squared loss between the reference function and the parameterized single feature function over given unsupervised data $\mathcal{D}$;

$$
\begin{aligned}
(\hat{\mathbf{u}}, \hat{\boldsymbol{\delta}}) &= \arg\min_{\mathbf{u}, \boldsymbol{\delta}} \left\{ \mathcal{O}(\mathbf{u}, \boldsymbol{\delta}; \mathcal{D}) \right\} \\
\mathcal{O}(\mathbf{u}, \boldsymbol{\delta}; \mathcal{D}) &= \frac{1}{2} \sum_i \sum_{(\mathbf{x}, y)} \left(\tilde{r}(\mathbf{x}, y) - u_i \delta_i f_i(\mathbf{x}, y) \right)^2 \\
&\quad + \lambda_1 ||\mathbf{u}||_1 + \frac{\lambda_2}{2} ||\mathbf{u}||_2^2 \\
\text{subject to:} \quad & \mathbf{u} \geq \mathbf{0},\ \boldsymbol{\delta} \in \{-1, 1\}^{|\mathcal{F}|},\ \mathbf{u} \in \mathcal{S}_K^{|\mathcal{F}|},
\end{aligned}
\tag{1}
$$

where the solution is represented as $\hat{\mathbf{u}} = (\hat{u}_i)_{i=1}^{|\mathcal{F}|}$, and $\hat{\boldsymbol{\delta}} = (\hat{\delta})_{i=1}^{|\mathcal{F}|}$, and $\sum_{(\mathbf{x}, y)}$ is an abbreviation of $\sum_{\mathbf{x} \in \mathcal{D}} \sum_{y \in \mathcal{Y}(\mathbf{x})}$. An intuitive interpretation of the solution $(\hat{\mathbf{u}}, \hat{\boldsymbol{\delta}})$ to this optimization problem is the 'impact' of each feature on the decision $\tilde{r}(\cdot)$ evaluated by the initial model. $\hat{\delta}_i = 1$ is obtained if $f_i(\cdot)$ is correlated with the positive decision $\tilde{r}(\cdot) > 0$, and $\hat{\delta}_i = -1$ if $f_i(\cdot)$ is correlated with the negative decision $\tilde{r}(\cdot) < 0$. Moreover, $\hat{u}_i$ will take a large value if $f_i(\cdot)$ has a large impact on whether a decision is positive or negative.

Build Fused Feature Set after Optimization This section describes how to generate a fused feature set $\mathcal{H}$ automatically from the solution of the optimization $(\hat{\mathbf{u}}, \hat{\boldsymbol{\delta}})$ explained in Eq. 1. By the effect of the constraint $\mathbf{u} \in \mathcal{S}_K^{|\mathcal{F}|}$ in Eq. 1, the number of unique values in $(\hat{u}_i)_{i=1}^{|\mathcal{F}|}$ except for zero is K (or less than K). Therefore, following Def. 2, we assign the unique values of $(\hat{u}_i)_{i=1}^{|\mathcal{F}|}$ sorted in descending order to $\{v_k\}_{k=1}^{K}$ except for zero. Next, we assign $f_i(\cdot)$ to $\mathcal{F}_k$ where their corresponding values, $\hat{u}_i$ and v_k, are equivalent. Namely, $\mathcal{F}_k$ can be written as: $\mathcal{F}_k = \{f_i(\cdot) | f_i(\cdot) \in \mathcal{F}, \hat{u}_i = v_k\}$. Now, we can generate $\mathcal{H}$ since we can simply obtain $\mathcal{F}_k$ and v_k from $\hat{\mathbf{u}}$. Fig. 1 shows an example where $K = 2$.

Overall, our optimization process can be interpreted as jointly weighting, selecting and grouping features via optimization in terms of the impact of each feature. Then, our method fuses all the features at the same impact level into one new feature. The intuition behind the fused feature representation is that many solution variables of standard supervised learning often take very similar values when modeling a target task with a large feature set. Given this fact, we assume that no or limited negative effects occur even if we fuse the features that have 'similar impact' in terms of the modeling of the target task.

Input: data:$\mathcal{D}$, tunable parameters:ρ, η, ϵ_{primal}, ϵ_{dual}
Initialize: $\mathbf{u}^{(1)} = \mathbf{0}$, $\boldsymbol{\delta}^{(1)} = \mathbf{1}$, $\mathbf{p}^{(1)} = \mathbf{0}$, $\boldsymbol{\alpha}^{(1)} = \mathbf{0}$, and $t = 1$.
Step1:
$$(\mathbf{u}^{(t+1)}, \boldsymbol{\delta}^{(t+1)}) = \arg\min_{(\mathbf{u}, \boldsymbol{\delta})} \left\{ \mathcal{O}(\mathbf{u}, \boldsymbol{\delta}; \mathcal{D}, \mathbf{p}^{(t)}, \boldsymbol{\alpha}^{(t)}) \right\} \tag{3}$$
Step2:
$$\mathbf{p}^{(t+1)} = \arg\min_{\mathbf{p}} \{ \mathcal{O}(\mathbf{p}; \mathbf{u}^{(t+1)}, \boldsymbol{\delta}^{(t+1)}, \boldsymbol{\alpha}^{(t)}) \} \tag{4}$$
Step3:
$$\boldsymbol{\alpha}^{(t+1)} = \boldsymbol{\alpha}^{(t)} + \rho\eta(\mathbf{u}^{(t+1)} - \mathbf{p}^{(t+1)}) \tag{5}$$
Step4: return if the convergence conditions are satisfied, or $t \leftarrow t + 1$ and go to Step1 otherwise.
Output: $(\mathbf{p}^{(t)}, \boldsymbol{\delta}^{(t)})$

Figure 2: Entire optimization framework of our method based on ADMM (Boyd et al. 2011)

Reformulation by Dual Decomposition

The previous section detailed our problem definition. This section explains how to solve Eq. 1. The main consideration is that Eq. 1 is essentially a combinatorial optimization problem because of the constraints $\boldsymbol{\delta} \in \{-1, 1\}^{|\mathcal{F}|}$ and $\mathbf{u} \in \mathcal{S}_K^{|\mathcal{F}|}$. Thus, a naive algorithm may take exponential time against n to obtain the solution. To derive a tractable optimization algorithm, we leverage the dual decomposition and proximal gradient techniques developed rapidly in recent years by the machine learning community *i.e.*, (Beck and Teboulle 2009; Boyd et al. 2011). The important property in our case is that it may allow us to decompose the intractable optimization problem into two (or many) tractable sub-problems, which are easily solvable. For example, (Zhong and Kwok 2012) showed an interesting example where a computationally hard combinatorial optimization problem in an optimization with OSCAR regularizers can be vanished by decomposition derived with the help of proximal gradient methods. We follow this idea, and reformulate Eq. 1 by using the dual decomposition technique (Everett 1963):

$$
\begin{aligned}
(\hat{\mathbf{u}}, \hat{\boldsymbol{\delta}}) &= \arg\min_{\mathbf{u}, \boldsymbol{\delta}} \left\{ \mathcal{O}(\mathbf{u}, \boldsymbol{\delta}; \mathcal{D}) \right\} \\
\mathcal{O}(\mathbf{u}, \boldsymbol{\delta}; \mathcal{D}) &= \frac{1}{2} \sum_i \sum_{(\mathbf{x}, y)} \left(\tilde{r}(\mathbf{x}, y) - u_i \delta_i f_i(\mathbf{x}, y) \right)^2 \\
&\quad + \lambda_1 ||\mathbf{u}||_1 + \frac{\lambda_2}{2} ||\mathbf{u}||_2^2 \\
\text{subject to:} \quad & \mathbf{u} \geq \mathbf{0}, \boldsymbol{\delta} \in \{-1, 1\}^{|\mathcal{F}|}, \mathbf{u} = \mathbf{p}, \mathbf{p} \in \mathcal{S}_K^{|\mathcal{F}|}.
\end{aligned}
\tag{2}
$$

To solve the optimization in Eq. 2, we leverage ADMM similarly to (Yang et al. 2012). Here, $\boldsymbol{\alpha}$ represents dual variables (or Lagrangian multipliers) for the equivalence constraint $\mathbf{u} = \mathbf{p}$. The optimization problem in Eq. 2 can be converted into a series of iterative optimization problems. A detailed general derivation of ADMM in a general case can be found in (Boyd et al. 2011). Fig. 2 shows the entire optimization framework based on ADMM for Eq. 2. ADMM works by iteratively computing one of the three optimization variable sets $\mathbf{u}$, $\mathbf{p}$, and $\boldsymbol{\alpha}$ while holding the other parameters fixed in the iterations $t = 1, 2, \ldots$ until convergence.

ADMM Step1, namely, Eq. 3 in Fig. 2 derived for Eq. 2, can be written as follows:

$$(\hat{\mathbf{u}}, \hat{\boldsymbol{\delta}}) = \arg\min_{(\mathbf{u},\boldsymbol{\delta})} \left\{\mathcal{O}(\mathbf{u}, \boldsymbol{\delta}; \mathcal{D}, \mathbf{p}, \boldsymbol{\alpha})\right\}$$
$$\begin{aligned}\mathcal{O}(\mathbf{u}, \boldsymbol{\delta}; \mathcal{D}, \mathbf{p}, \boldsymbol{\alpha}) = \\ \frac{1}{2}\sum_i \sum_{(\mathbf{x},y)} \left(\tilde{r}(\mathbf{x}, y) - u_i \delta_i f_i(\mathbf{x}, y)\right)^2 \\ +\lambda_1 ||\mathbf{u}||_1 + \frac{\lambda_2}{2}||\mathbf{u}||_2^2 + \boldsymbol{\alpha}(\mathbf{u} - \mathbf{p}) + \frac{\rho}{2}(\mathbf{u} - \mathbf{p})^2 \\ \text{subject to:} \quad \mathbf{u} \geq \mathbf{0},\ \boldsymbol{\delta} \in \{-1, 1\}^{|\mathcal{F}|}.\end{aligned} \tag{6}$$

Then, ADMM Step2 of Eq. 4 in Fig. 2 for our case can be written in this form:

$$\begin{aligned}\hat{\mathbf{p}} = \arg\min_{\mathbf{p}}\{\mathcal{O}(\mathbf{p}; \mathbf{u}, \boldsymbol{\alpha})\} \\ \mathcal{O}(\mathbf{p}; \mathbf{u}, \boldsymbol{\alpha}) = \frac{\rho}{2}\sum_i \left(p_i - q_i\right)^2 \\ \text{subject to:} \quad \mathbf{p} \in \mathcal{S}_K^{|\mathcal{F}|}.\end{aligned} \tag{7}$$

where $q_i = u_i + \frac{\alpha_i}{\rho}$. Note that ADMM Step3 in Fig. 2 is identical to the original step for our case.

Basic Optimization Algorithm

So, to solve our problem Eq. 1, we must find an efficient way to solve Eqs. 6 and 7. This section reveals that Eqs. 6 and 7 can be solved by linear and polynomial time algorithms with no iterative estimations.

Step1: $(\mathbf{u}, \boldsymbol{\delta})$-update (Gradient Step) The optimal solution of Eq. 6, $(\hat{\mathbf{u}}, \hat{\boldsymbol{\delta}})$, should hold $\partial_{u_i}\mathcal{O}(\hat{\mathbf{u}}, \hat{\boldsymbol{\delta}}; \mathcal{D}, \mathbf{p}, \boldsymbol{\alpha}) = 0$, that is, the partial derivatives of the objective in Eq. 6 with respect to u_i equal to zero. Since $\partial_{u_i}\mathcal{O}(\mathbf{u}, \boldsymbol{\delta}; \mathcal{D}, \mathbf{p}, \boldsymbol{\alpha}) = -\sum_{(\mathbf{x},y)}(\tilde{r}(\mathbf{x}, y) - u_i\delta_i f_i(\mathbf{x}, y))(\delta_i f_i(\mathbf{x}, y)) + \lambda_1 + \lambda_2 u_i + \alpha_i + \rho(u_i - p_i)$, we obtain the following closed form:

$$\begin{aligned}u_i = \frac{\delta_i R_i - \lambda_1 - \alpha_i + \rho p_i}{F_i + \lambda_2 + \rho}, \text{ where} \\ F_i = \sum_{(\mathbf{x},y)} f_i(\mathbf{x}, y), \text{ and } R_i = \sum_{(\mathbf{x},y)} \tilde{r}(\mathbf{x}, y) f_i(\mathbf{x}, y).\end{aligned} \tag{8}$$

F_i and R_i are obtained by the facts $(\delta_i)^2 = 1$ and $(f_i(\mathbf{x}, y))^2 = f_i(\mathbf{x}, y)$ derived from Def. 4 and Assumption 3. Moreover, $F_i + \lambda_2 + \rho > 0$ always holds from the definitions of $F_i > 0$, $\lambda_2 \geq 0$ and $\rho > 0$ for the denominator of u_i in Eq. 8. Therefore, if the numerator of u_i in Eq. 8 is equal to or less than 0, namely, $\delta_i R_i - \lambda_1 - \alpha_i + \rho p_i \leq 0$, then $u_i = 0$ always holds because of the constraint $\mathbf{u} \geq \mathbf{0}$. Therefore, the optimal solution $\hat{\delta}_i$ can be easily selected[2]:

$$\hat{\delta}_i = \begin{cases} +1 & \text{if } R_i \geq 0, \\ -1 & \text{otherwise} \end{cases}. \tag{9}$$

let $[a]_+ = \max(0, a)$. Then, the optimal solution of $\hat{u}_i$ can also be obtained by using $\hat{\delta}_i$:

$$\hat{u}_i = \left[\frac{\hat{\delta}_i R_i - \lambda_1 - \alpha_i + \rho p_i}{F_i + \lambda_2 + \rho}\right]_+. \tag{10}$$

[2]Note that if $\delta_i R_i - \lambda_1 - \alpha_i + \rho p_i \leq 0$ holds for both $\delta_i = 1$ and $\delta_i = -1$, then the selection of δ_i has no effect on the objective. This means that there are several optimal solutions that all give the same objective values. An important point is that Eq. 9 provides at least one of the optimal solutions.

As a result, (one of) the optimal solutions $(\hat{\mathbf{u}}, \hat{\boldsymbol{\delta}})$ can be obtained in a closed form. This indicates that we can always obtain an exact solution for the optimization of Eq. 6 without an iterative estimation, whose time complexity is $O(|\mathcal{F}|)$.

Step2: p-update (Projection Step) This section presents an algorithm for obtaining the optimal solution of Step2. According to the relations $p_i = v \in \mathcal{S}_K$ for all i, Eq. 7 can be rewritten in the following equivalent simple form:

$$\hat{\mathcal{S}}_K = \arg\min_{v \in \mathcal{S}_K} \frac{1}{2}\sum_i (v - q_i)^2, \tag{11}$$

Note that this equation and Eq. 7 are equivalent problems. We can easily recover $\hat{\mathbf{p}}$ from $\hat{\mathcal{S}}_K$ by selecting the nearest value in $\hat{\mathcal{S}}_K$ from q_i for each $\hat{p}_i$. Eq. 11 is also known as a one-dimensional K-means clustering problem (Wang and Song 2011). Although one-dimensional K-means clustering is still a combinatorial optimization problem, a dynamic programming based polynomial time exact algorithm called *Ckmeans.1d.dp*, is introduced in (Wang and Song 2011), whose time and space complexities are $O(KM^2)$ and $O(KM)$, respectively. Note that, in our case, $K = |\mathcal{H}|$ and $M = |\mathcal{F}|$.

Step4: Convergence Check For the convergence check for Step4, we evaluate both primal and dual residuals as defined in (Boyd et al. 2011), that is, $\frac{1}{|\mathcal{F}|}||\mathbf{u}^{(t+1)} - \mathbf{p}^{(t+1)}||^2 < \epsilon_{\text{primal}}$ and $\frac{1}{|\mathcal{F}|}||\boldsymbol{\alpha}^{(t+1)} - \boldsymbol{\alpha}^{(t)}||^2 < \epsilon_{\text{dual}}$, with suitably small ϵ_{primal} and ϵ_{dual}. To force the optimization to always converge, we gradually increase ρ from 1 to ∞ in every iteration. This brings Eq. 10 closer to $\hat{u}_i = p_i$. At this time, it is obvious that the above primal and dual residuals both become 0. Finally, ADMM outputs $(\mathbf{p}^{(t)}, \boldsymbol{\delta}^{(t)})$ as a solution[3].

Time and Space Complexities We iteratively calculate Step1 through Step4 shown in Fig. 2 to solve Eq. 1. Let T be the number of iterations. The time and space complexities for calculating Step1 are $O(|\mathcal{D}||\mathcal{F}|)$ and $O(|\mathcal{F}|)$, respectively. Similarly, $O(|\mathcal{H}||\mathcal{F}|^2)$ and $O(|\mathcal{H}||\mathcal{F}|)$ for Step2, and $O(|\mathcal{F}|)$ for both for Step3. Hence, the total time complexity of our algorithm is $O(T(|\mathcal{D}||\mathcal{F}| + |\mathcal{H}||\mathcal{F}|^2))$.

Extensions for Large Data and Feature Sets

The previous section introduced the basic optimization algorithm. This section focuses on describing the several important properties that allow us to substantially reduce the computational cost, especially with large problems.

For Large Data sets

Property 1: *Every gradient step can be calculated without re-evaluation of data points.*

proof: F_i and R_i can be calculated without $\mathbf{u}$. This means that it is not necessary to retain any data points during the entire optimization process if we calculate and cache F_i and R_i for all i before starting the optimization and use them instead of the data point information. □

[3]Note that $\mathbf{u}^{(t)}$ is unnecessary since we substitute $\mathbf{p}^{(t)}$ for $\mathbf{u}^{(t)}$.

As a result, the time complexity is reduced from $O(T(|\mathcal{D}||\mathcal{F}|+|\mathcal{H}||\mathcal{F}|^2))$ to $O(|\mathcal{D}||\mathcal{F}|+T|\mathcal{H}||\mathcal{F}|^2)$. This reduction is extremely important for handling large data sets, *i.e.*, when $|\mathcal{D}|$ is millions or billions.

Property 2: *The calculations of F_i and R_i for all i can be decomposed into data-point-wise calculations.*

proof: Eq. 8 showed that F_i and R_i consist only of the i-th feature and reference functions. Moreover, F_i and R_i take the form of a linear combination for every data point. □

This property readily suggests the use of parallel computation. In fact, the F_i and R_i calculations are suitable for the MapReduce model (Dean and Ghemawat 2008). Note that F_i and R_i can be calculated in a single Map-Reduce operation. This reduces the time complexity to $O(\frac{|\mathcal{D}||\mathcal{F}|}{P} + T|\mathcal{H}||\mathcal{F}|^2)$, where P is the parallelization number.

For Large Feature Sets

Property 3: *The upper bound number of non-zero optimization parameters in the optimal solution can be estimated before starting the optimization.*

proof: According to Eq. 10, $\hat{u}_i = 0$ always holds if $\hat{\delta}_i R_i - \lambda_1 \leq 0$, since λ_1 and $\hat{\delta}_i R_i$ are both non-negative real value constants during the optimization process. Moreover, $\alpha_i = 0$ and $p_i = 0$ always hold in this situation since $\hat{u}_i = 0$ always holds for the solution of Step1. Thus, we can first judge whether $\hat{u}_i$ will become zero, or possibly non-zero, before starting the optimization process by evaluating whether or not $\hat{\delta}_i R_i - \lambda_1 \leq 0$ is satisfied. □

This property reveals that we can control the maximum number of original features that map to fused features before starting the optimization process by controlling the λ_1 setting. In a real implementation, we start by sorting $|R_i|$ for all i in descending order, and then set λ_1 as the $(|\bar{\mathcal{F}}| + 1)$-st largest $|R_i|$ if we want to restrict the $|\bar{\mathcal{F}}|$ parameters so that they are non-zero at most. This procedure can also reduce the computational cost. This is because if we already know that $\hat{u}_i = 0$, we can drop parameter u_i from the optimization. Let $|\bar{\mathcal{F}}|$ be the number of parameters that are possibly non-zero, where $|\bar{\mathcal{F}}| \leq |\mathcal{F}|$. The time complexity becomes $O(\frac{|\mathcal{D}||\mathcal{F}|}{P} + T|\mathcal{H}||\bar{\mathcal{F}}|^2)$. Note that $\hat{u}_i = 0$ occurs even if $|R_i| > \lambda_1$ because of the effect of the procedure for Step2.

In addition, by controlling λ_1, we can possibly handle an infinite number of original features. Our method only considers top-$|\bar{\mathcal{F}}|$ features. Therefore, there is no need to define the total number of features.

Lower complexity 1D K-means clustering In actual use, the $O(KM^2)$ time complexity is insufficient when M is very large *i.e.*, $M = |\mathcal{F}| > 1,000,000$. We attempt to modify the algorithm to reduce the time complexity.

Proposition 6. *Let $\mathbf{q}' = (q'_1, \ldots, q'_M)$ be $\mathbf{q}$ sorted in the ascending order. Let $c_{i,j}$ represent a cluster consisting of the data points from q'_i to q'_j, and $\bar{\mu}_{i,j}$ be its centroid. Assume that we obtain the optimal cluster set $(\hat{c}_{1,i}, \ldots, \hat{c}_{j,M})$ of 1D K-means clustering. Moreover, we define $\bar{\mu}_{i,j,k} = \frac{1}{2}(\bar{\mu}_{i,j-1} + \bar{\mu}_{j,k})$. Then, q'_{j-1} and q'_j, which are the rightmost or leftmost data points of two adjacent optimal clusters $\hat{c}_{i,j-1}$ and $\hat{c}_{j,k}$, always maintain the following relation; $q'_{j-1} \leq \bar{\mu}_{i,j,k} \leq q'_j$.*

proof: $\bar{\mu}_{i,j,k}$ essentially represents the middle of two adjacent optimal cluster centroids. If $\bar{\mu}_{i,j,k} < q'_{j-1}$, then the point q'_{j-1} has to be included in the cluster $\hat{c}_{j,k}$ since $(q'_{j-1} - \bar{\mu}_{i,j-1})^2 > (q'_{j-1} - \bar{\mu}_{j,k})^2$. Similarly, if $q'_j < \bar{\mu}_{i,j,k}$ then q'_{j-1} has to be included in the cluster $\hat{c}_{i,j-1}$. Therefore, in these two cases, the optimal clustering condition is violated. In contrast, if $q'_{j-1} \leq \bar{\mu}_{i,j,k} \leq q'_j$ holds, then $(q'_{j-1} - \bar{\mu}_{i,j-1})^2 \leq (q'_{j-1} - \bar{\mu}_{j,k})^2$, and $(q'_j - \bar{\mu}_{i,j-1})^2 \geq (q'_j - \bar{\mu}_{j,k})^2$ are always satisfied. □

We can significantly speed up Step2 by using Prop. 6. We introduce a binary search procedure. This procedure simply attempts to find a point q'_k that satisfies the condition in Prop. 6, that is, $q'_{j-1} \leq q'_k \leq q'_j$, and does not perform the calculation if q'_k does not satisfy the above relation. As a result, the total time complexity is reduced to $O(KM \log M)$ from $O(KM^2)$. The time complexity finally becomes $O(\frac{|\mathcal{D}||\mathcal{F}|}{P} + T|\mathcal{H}||\bar{\mathcal{F}}| \log |\bar{\mathcal{F}}|)$.

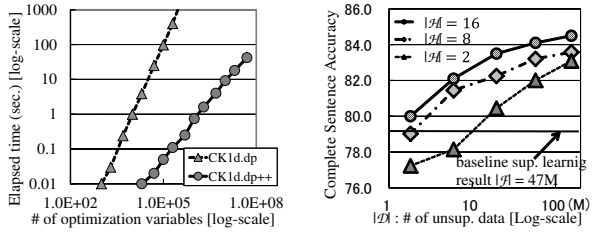
Experiments

We conducted experiments on the data sets of two well-studied natural language processing (NLP) tasks, namely named entity recognition (NER) and dependency parsing (DEPAR). Wee simply followed the experimental settings described in previous studies providing state-of-the-art results, *i.e.*, (Suzuki and Isozaki 2008) for NER, and (Koo, Carreras, and Collins 2008) for DEPAR.

We gathered 150 million data points (sentences) from the LDC corpus, that is, $|\mathcal{D}|$=150M. The total numbers of original feature sets appearing in $\mathcal{D}$ reached 20 and 6 billion, namely $|\mathcal{F}| = 20$B and $|\mathcal{F}| = 6$B, for NER and DEPER, respectively. These features are automatically generated by using canonical feature templates, which have been widely used in previous studies, over the $|\mathcal{D}| = 150$M unsupervised data. These feature sets are used as the input of our feature representation discovery. Similarly, the total numbers of original features that appeared in the supervised data, $\mathcal{D}_L$, were 47M for NER and 260M for DEPER. We selected CRF (Lafferty, McCallum, and Pereira 2001) for NER, and the online structured output learning version of the Passive-Aggressive algorithm (ostPA) (Crammer et al. 2006) for DEPAR as a supervised learning algorithm to build both the initial and final models.

In outline, our experimental procedure is as follows; (1) We build an initial model using $\mathcal{F}$ extracted only from $\mathcal{D}_L$. (2) We calculate F_i and R_i for all i over a MapReduce system. (3) We discard features according to Prop. 3 since 20 and 6 billion features are too many, and most of them are possibly redundant and useless. We retain 4.7M and 26M features, which amounts to 10% of $\mathcal{F}$ extracted only from $\mathcal{D}_L$, and all the remaining features are discarded from the input of our method. (4) We construct a fused feature set $\mathcal{H}$ using our method with various settings of K. (5) We build a final model using $\mathcal{H}$.

Finally, we evaluate both the initial and final models. The task performance was evaluated in terms of complete sen-



(a) (b)

Figure 3: (a): Relation between elapsed time and number of optimization variables in Step2 when $K = 8$. (b): Learning curve against the quantity of unsupervised data.

tence accuracy (**COMP**), the $F_{\beta=1}$ score (**F-sc**) for NER, and the unlabeled attachment score (**UAS**) for DEPAR for comparison with previous studies.

Efficiency of proposed algorithms We estimate that it would take roughly 5,877 hours (approximately 245 days) to calculate a single gradient step for all the DEPAR data (we did not attempt this). Obviously, this is unacceptable. On the other hand, the total run time of our method on the DEPAR data was, as one example, about 15 hours, where approximately 14 hours were taken for calculating the gradient step with 512 nodes of our MapReduce system (Property 2), and the additional 1 hour was spent on the iterative estimation of Step1 through Step4. Note that our method requires only one costly gradient step calculation (Property 1). We believe this is a reasonable run time given the massive quantity of data being processed.

In addition, we confirmed that the run times of $|\mathcal{H}|$ in the test phase were nearly equivalent but no more than those of $|\mathcal{F}|$. These results are essentially the positive results for showing the advantage of our fused feature representation.

Effectiveness of Prop. 6 The remaining time consuming part of our algorithm against $|\mathcal{F}| = M$ is Step2. Fig. 3 (a) shows the effectiveness of our algorithm based on Prop. 6 (**CK1d.dp++**). For example, our algorithm took less than 0.1 sec for 100K variables while the original algorithm (Wang and Song 2011) took 97.24 sec (**CK1d.dp**). Moreover, CK1d.dp++ successfully worked with 10M variables within 10 sec. It is worth noting here that, empirically, the actual elapsed time of our modified algorithm had a near linear relation against the M increase, although the time complexity was $O(KM \log M)$.

Learning curve Fig. 3 (b) shows the learning curves of our method with respect to the number of unsupervised data in our NER experiments. The x-axis shows the logarithmic values of unsupervised data size (the scale is millions). Each line indicates the performance change when we increased the unsupervised data in a situation where we fixed the number of fused features obtained with our method. It is clear that better performance is achieved as the unsupervised data size increases in any setting. This constitutes good evidence for the importance of handling as much data as possible to achieve performance improvement, and the importance of using a feasible algorithm with the large data set.

Table 1: Comparison with previous top-line systems.

NER system	COMP	F-sc	$\lvert\mathcal{D}\rvert$	$\lvert\mathcal{F}\rvert$	$\lvert\mathcal{H}\rvert$
supervised (L_2-CRF)	79.10	85.08	0	47M	–
(Suzuki and Isozaki 2008)	N/A	89.92	≈50M	N/A	–
(Lin and Wu 2009)	N/A	**90.90**	≈30B	N/A	–
our method	**84.50**	89.04	150M	20B	16
	83.58	88.78	150M	20B	8
	83.09	88.39	150M	20B	2

Dependency parser	COMP	UAS	$\lvert\mathcal{D}\rvert$	$\lvert\mathcal{F}\rvert$	$\lvert\mathcal{H}\rvert$
supervised (ostPA)	47.60	92.82	0	260M	–
(Martins et al. 2010)	N/A	93.26	0	55M	–
(Koo et al. 2008)	N/A	93.16	≈2.5M	N/A	–
(Suzuki et al. 2009)	N/A	93.79	≈150M	N/A	–
(Chen et al.2013)	**51.36**	93.77	≈2.5M	27M	–
our method	50.58	**93.85**	150M	6B	16
	49.71	93.68	150M	6B	8
	48.22	92.90	150M	6B	2

Quality evaluation Table 1 summarizes the performance of our method and the current top-line NER and DEPAR systems constructed by supervised (first row) and semi-supervised learning (second row). We emphasize that the numbers of features obtained by our method were much smaller than those of the original feature sets. These results indicate that our method has the ability to provide meaningful low-dimensional feature representations from extremely large data and feature sets. Interestingly, our method also matched or even outperformed the baseline for $|\mathcal{H}| = 2$. These results revealed that it is possible to obtain a state-of-the-art performance with a very small number of parameters if we can build an appropriate feature set.

In addition, our method achieved the same level of results as a recently developed top-line semi-supervised learning system. We believe that our method is a promising approach for real applications since it is very simple, and highly suited to handling larger data sets.

Conclusion

This paper discussed feature representation discovery problems with a specific focus on extremely large data and feature sets. Our method basically consists of three parts. The first is a novel definition of reduced feature representation, which we call fused feature representation. The second is a definition of the problem, which essentially evaluates the impact of each feature on modeling the target task. The third is the addition of a clustering constraint to the optimization process, which realizes the joint weighting, selection and grouping of the features via optimization. We also introduced a tractable algorithm for large problems; its time complexity was finally reduced to $O(\frac{|\mathcal{D}||\mathcal{F}|}{P} + T|\mathcal{H}||\bar{\mathcal{F}}| \log |\bar{\mathcal{F}}|)$. The experimental results were promising, and provided several instances that confirmed the impact of feature representation discovery algorithms tuned for large-scale problems such as our method.

Acknowledgement

We thank three anonymous reviewers for their helpful comments.

References

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

Boyd, S.; Parikh, N.; Chu, E.; Peleato, B.; and Eckstein, J. 2011. *Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipliers*. Foundations and Trends in Machine Learning.

Chen, W.; Zhang, M.; and Zhang, Y. 2013. Semi-supervised feature transformation for dependency parsing. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, 1303–1313. Seattle, Washington, USA: Association for Computational Linguistics.

Crammer, K.; Dekel, O.; Keshet, J.; Shalev-Shwartz, S.; and Singer, Y. 2006. Online Passive-Aggressive Algorithms. *Journal of Machine Learning Research* 7:551–585.

Dean, J., and Ghemawat, S. 2008. MapReduce: Simplified Data Processing on Large Clusters. *Communications of the ACM* 51(1):107–113.

Druck, G., and McCallum, A. 2010. High-performance Semi-Supervised Learning using Discriminatively Constrained Generative Models. In *Proceedings of the International Conference on Machine Learning (ICML 2010)*, 319–326.

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

Hinton, G. E. 2007. Learning Multiple Layers of Representations. *Trends in Cognitive Sciences* 11(10):428–434.

Koo, T.; Carreras, X.; and Collins, M. 2008. Simple Semi-supervised Dependency Parsing. In *Proceedings of ACL-08: HLT*, 595–603.

Kuksa, P. P., and Qi, Y. 2010. Semi-supervised Bio-named Entity Recognition with Word-codebook Learning. In *Proceedings of the SIAM International Conference on Data Mining (SDM)*, 25–36.

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

Lin, D., and Wu, X. 2009. Phrase Clustering for Discriminative Learning. 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*, 1030–1038.

Martins, A.; Smith, N.; Xing, E.; Aguiar, P.; and Figueiredo, M. 2010. Turbo Parsers: Dependency Parsing by Approximate Variational Inference. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, 34–44.

Suzuki, J., and Isozaki, H. 2008. Semi-supervised Sequential Labeling and Segmentation Using Giga-Word Scale Unlabeled Data. In *Proceedings of ACL-08: HLT*, 665–673.

Suzuki, J.; Isozaki, H.; Carreras, X.; and Collins, M. 2009. An Empirical Study of Semi-supervised Structured Conditional Models for Dependency Parsing. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, 551–560.

Turian, J.; Ratinov, L.-A.; and Bengio, Y. 2010. Word Representations: A Simple and General Method for Semi-Supervised Learning. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, 384–394.

Van Der Maaten, L., and Hinton, G. 2008. Visualizing data using t-SNE. *Journal of Machine Learning Research (JMLR)* 9:2579–2605.

Wang, H., and Song, M. 2011. Ckmeans.1d.dp: Optimal k-means Clustering in One Dimension by Dynamic Programming. *The R Journal* 3(2):29–33.

Yang, S.; Yuan, L.; Lai, Y.-C.; Shen, X.; Wonka, P.; and Ye, J. 2012. Feature Grouping and Selection over an Undirected Graph. In *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 922–930.

Zhang, X.; Yu, Y.; White, M.; Huang, R.; and Schuurmans, D. 2011. Convex Sparse Coding, Subspace Learning, and Semi-Supervised Extensions. In *Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence (AAAI)*, 567–573.

Zhong, L. W., and Kwok, J. T. 2012. Efficient sparse modeling with automatic feature grouping. *IEEE Transaction on Neural Networks and Learning Systems* 23(9):1436–1447.