

Document Clustering with Universum

Dan Zhang
Computer Science
Department
Purdue University,
West Lafayette, IN 47907, US
zhang168@cs.purdue.edu

Jingdong Wang
Microsoft Research Asia
Beijing, P.R. China, 100080
jingdw@microsoft.com

Luo Si
Computer Science
Department
Purdue University,
West Lafayette, IN 47907, US
lsi@cs.purdue.edu

ABSTRACT

Document clustering is a popular research topic, which aims to partition documents into groups of similar objects (i.e., clusters), and has been widely used in many applications such as automatic topic extraction, document organization and filtering. As a recently proposed concept, Universum is a collection of “non-examples” that do not belong to any concept/cluster of interest. This paper proposes a novel document clustering technique – Document Clustering with Universum, which utilizes the Universum examples to improve the clustering performance. The intuition is that the Universum examples can serve as supervised information and help improve the performance of clustering, since they are known not belonging to any meaningful concepts/clusters in the target domain. In particular, a maximum margin clustering method is proposed to model both target examples and Universum examples for clustering. An extensive set of experiments is conducted to demonstrate the effectiveness and efficiency of the proposed algorithm.

Categories and Subject Descriptors

H.3.3 [Information Storage and Retrieval]: Information Search and Retrieval—*clustering*; I.2.6 [Artificial Intelligence]: Learning

General Terms

Algorithms, Performance, Experimentation

Keywords

Clustering, Universum, Maximum Margin Clustering, Constrained Concave-Convex Procedure (CCCP)

1. INTRODUCTION

Document clustering is a very important topic in information retrieval, and has received substantial attentions for

unsupervised document organization, automatic topic extraction, etc (e.g. [1, 2, 7, 13, 15, 21, 23]). It aims to partition documents into groups of similar objects (i.e., clusters). From the information retrieval perspective, what document clustering does is to learn hidden patterns underlying the available documents, where these hidden patterns may contain some interesting concepts. A good clustering algorithm can automatically organize documents into a set of meaningful clusters for efficient browsing and navigation.

Current document clustering algorithms can be roughly divided into two categories (i.e., hierarchical methods and partitioning methods). The hierarchical methods divide the data into a hierarchical structure by employing bottom up or top down methods [6, 15, 18]. A typical algorithm for the hierarchical method is the hierarchical agglomerative clustering, which starts from taking each document as a single cluster, and then merges these documents gradually to build larger and larger clusters, until the whole dataset is merged as one single cluster. Partitioning methods divide the whole dataset into a fixed number of disjoint clusters [5]. A typical algorithm for the partitioning method is KMeans [5]. The basic motivation of KMeans is to find a set of clustering assignments such that the sum of distances between examples and their associated cluster centers can be minimized. Besides KMeans, some probabilistic methods such as Gaussian Mixture Model (GMM) [14] and Probabilistic Latent Semantic Analysis (PLSA) [8], as well as some graph based methods such as Normalized Cut (NCut) [19], have also shown promising results in clustering. However, one of the most significant drawbacks of these methods is that they only consider the target examples that need to be partitioned, but neglect other information that could be beneficial for clustering. It makes the clustering results only depend on the target examples. As shown in Fig.1(a), the traditional partitioning method wrongly groups one data point (square) into the other cluster, without employing any other prior knowledge.

Universum is a set of examples that do not belong to any concept/cluster of interest, which are different from the target examples that we need to cluster. Some previous work [4, 20, 28, 35] has demonstrated the benefits of treating these examples as prior knowledge imposed on the classification hyperplanes. However, there is no work on applying the Universum examples to the clustering problems. The major contribution of this paper is to show that Universum examples can be used to improve the performance of document clustering. The intuition is that Universum examples should not have a clear pattern, or meaningful concepts/clusters in

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

SIGIR '11, July 24–28, 2011, Beijing, China.

Copyright 2011 ACM 978-1-4503-0757-4/11/07 ...\$10.00.

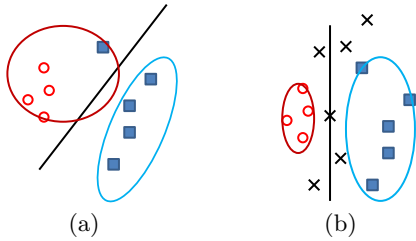


Figure 1: (a) An example clustering result of a typical clustering algorithm (partitioning) with two classes of examples (i.e., circles for class 1 and squares for class 2). It is clear that one square is misclassified; (b) An example clustering result with the guidance of Universum examples (i.e., check marks). Since Universum examples should not have a clear pattern/meaningful cluster to belong to, they should be distributed around the hyperplane, and provide some supervised information for the clustering problem. As can be seen from this picture, the previous misclassified example is now correctly classified.

the target domain. This intuition provides valuable guidance of the clustering process. As shown in Fig.1(b), since the Universum examples do not belong to any of the two clusters, it should only distribute around the hyperplane that separates these two clusters. With the guidance of these Universum examples, the square that was mis-clustered in Fig.1(a) can now be assigned to the right group.

This paper proposes a novel *Document Clustering with Universum (DCU)* problem, which utilizes Universum examples in document clustering. In particular, a formulation – *Maximum Margin Document Clustering with Universum (M^2DCU)* is presented to solve this problem. This formulation is based on Maximum Margin Clustering [30], and aims at finding desired hyperplanes that maximize the margins on the (target) examples that need to be partitioned, and minimize the margins on the Universum examples. The new formulation is not a convex optimization problem and therefore cannot be optimized directly. So, we first relax the original problem and then solve it by using *Constrained Concave-Convex Procedure (CCCP)* [26]. To accelerate the optimization speed for each sub optimization problem derived from CCCP, an extension of the *Cutting Plane* method [11] is designed. Our experiments on four real world datasets demonstrate the performance improvement brought by the Universum examples and show that the proposed method substantially outperforms state-of-the-art methods.

The rest of the paper is organized as: Section 2 introduces the related work. Section 3 puts forward the novel DCU problem. Section 4 proposes the M^2DCU algorithm. Section 5 presents the experimental results. Section 6 concludes this paper and points out some future research directions. The appendix proposes a novel optimization method to accelerate the optimization speed of the algorithm.

2. RELATED WORK

2.1 Universum

The concept of Universum was first proposed in solving classification problems, and was defined as a set of “non-examples” distributed in the same domain as the problem

of interest, but known not belonging to any classes that need to be classified [28]. The channels of obtaining Universum examples are very diverse. In [28], three simple ways are suggested, i.e., the examples from some other categories that are known unlikely to belong to any of the target categories; the randomly generated examples; as well as the examples generated by randomly combining examples from different categories. Since they are not likely to belong to any of the target categories, when designing classifiers, these Universum examples should not be classified to any of the categories either.

Out of this motivation, in [28], Jeston et al. designed the first classification algorithm that utilizes Universum examples – \mathcal{U} -Support Vector Machines (\mathcal{U} -SVM). This work adds a penalty term on the Universum examples to the formulation of the traditional Support Vector Machines (SVM) [5] so that the output given by the classifier will minimize the empirical classification loss on the target examples, and will not give clear classification assignments for the Universum examples. This work is reasonable and performs better than SVM, as shown in an extensive set of experiments [28]. However, it can only deal with binary supervised classification problems. Inspired by this paper, in [35], Zhang et al. extended the Universum algorithm to solve multi-class classification problems and improved the performance of semi-supervised learning by using Universum examples. Later, Huang et al. proposed to use the Universum examples in a different way to solve the semi-supervised classification problems [9]. In [20], the authors analyzed some properties of \mathcal{U} -SVM. In [4], Chen et al. designed a method to choose the most useful Universum examples for classification.

All of the previous work has contributed a lot to the use of Universum examples in classification problems. However, there is no prior work on how to use Universum examples in clustering problems. In fact, in clustering, it is natural to consider the Universum examples as examples that do not belong to any of the clusters in the target domain. In this way, the Universum examples can be viewed as some supervised information in an unsupervised problem. It is also interesting to notice that in the toy experiments of [20], \mathcal{U} -SVM performs much better than SVM when the number of labeled examples is small. In a clustering problem, since no labeled example is given, we would also expect a great advantage over the traditional clustering algorithms by the utilization of the Universum examples.

2.2 Maximum Margin Clustering (MMC)

The proposed method M^2DCU is based on a recently developed algorithm – *Maximum Margin Clustering (MMC)*. MMC was first proposed in [30]. In [30], the authors borrowed the idea of a standard machine learning principle - maximum margin principle, and used it for clustering. In particular, they tried to assign each of n instances to two classes $\{-1, +1\}$ so that the separation between the two classes can be as large as possible. It is formulated as:

$$\begin{aligned} \min_{\mathbf{y} \in \{-1, +1\}^n} \min_{\mathbf{w}, b, \xi_i} \quad & \frac{1}{2} \|\mathbf{w}\|^2 + \frac{C}{n} \sum_{i=1}^n \xi_i \\ \text{s.t.} \quad & \forall i = 1, \dots, n, \quad y_i(\mathbf{w}^T \mathbf{x}_i + b) \geq 1 - \xi_i, \\ & \xi_i \geq 0, -l \leq \mathbf{e}^T \mathbf{y} \leq l, \end{aligned} \quad (1)$$

where $\sum_{i=1}^n \xi_i$ is divided by the number of examples to better capture how C scales with the data set size, where C is a trade-off parameter for the two parts of this formulation.

$l \geq 0$ is a constant controlling the class imbalance and \mathbf{e} is an all-one vector [30]. The final clustering result is given by \mathbf{y} . It is clear that if we subsequently run an SVM with the clustering assignment obtained from MMC, the margin would be maximal among all possible labelings.

In this paper, we consider the distribution of Universum examples as prior knowledge in MMC. With similar motivations, in [29, 32], the authors propose to use some known similarity relationships (e.g., must-link, cannot-link and family/possible-link) between examples as the prior information in clustering. However, in their work, all the documents are still associated with some meaningful clusters, while this is not the case for clustering with Universum.

3. PROBLEM STATEMENTS

First of all, the new problem – Document Clustering with Universum (DCU) is formally proposed. Suppose we are given a set of documents/examples, represented by $\mathcal{X} = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n\}$, $\mathbf{x}_i \in \mathbb{R}^d$, where n is the total number of documents. Besides these documents, some Universum documents/examples, which are in the same domain as \mathcal{X} , but known not belonging to any meaningful groups in \mathcal{X} , are also available, and denoted as $\mathcal{X}^* = \{\mathbf{x}_1^*, \mathbf{x}_2^*, \dots, \mathbf{x}_N^*\}$, $\mathbf{x}_i^* \in \mathbb{R}^d$. N is the number of Universum examples. For convenience, throughout this paper, the examples in \mathcal{X} are denoted as “**target examples/documents**”, while the examples in \mathcal{X}^* are referred to as “**Universum examples/documents**”. The main objective of DCU is to divide the target examples into k clusters, with the help of Universum examples. A $1 \times n$ vector \mathbf{y} is used here to denote the cluster assignment array, with y_i being the cluster assignment for \mathbf{x}_i .

4. MAXIMUM MARGIN DOCUMENT CLUSTERING WITH UNIVERSUM

4.1 Formulation

The proposed formulation tries to find a set of hyperplanes that can separate examples from different clusters, and minimize the confidence of assigning Universum examples to any category. To model these hyperplanes, for each cluster $p \in \{1, 2, \dots, k\}$, we define a weight vector \mathbf{w}_p . For the target examples, we expect their cluster assignments to be as determined as possible. Therefore, M²DCU intends to maximize the clustering margins/confidence on the target examples. For the Universum examples, since they should not be assigned to any one of the k clusters, we expect that the margins¹/confidence on them should be small, which is expressed as:

$$\forall i = 1, \dots, N, \quad \mathbf{w}_{u_i^*}^T \mathbf{x}_i^* - \mathbf{w}_{v_i^*}^T \mathbf{x}_i^* \leq \xi_i^* + \epsilon_1, \quad (2)$$

where ϵ_1 is a tolerance parameter, $u_i^* = \arg \max_p \mathbf{w}_p^T \mathbf{x}_i^*$, $v_i^* = \arg \max_{p \setminus u_i^*} \mathbf{w}_p^T \mathbf{x}_i^*$ ². ξ_i^* is the nonnegative hinge loss that needs to be optimized on the Universum examples. The

¹It can also be considered as the difference between the largest output and the second largest output. As we shall see later, this kind of margin definition on Universum examples is also consistent with the margin defined on the target examples in Eq.(7).

²Throughout this paper, “\” means ruling out.

concrete formulation of M²DCU can be written as:

$$\min_{\mathbf{w}_p, \mathbf{y}, \xi_i \geq 0, \xi_i^* \geq 0} \quad \frac{1}{2} \sum_{p=1}^k \|\mathbf{w}_p\|^2 + \frac{C_l}{n} \sum_{i=1}^n \xi_i + \frac{C_u}{N} \sum_{i=1}^N \xi_i^* \quad (3)$$

$$s.t. \quad \forall i = 1, \dots, n, r = 1, \dots, k \quad (4)$$

$$\mathbf{w}_{y_i}^T \mathbf{x}_i + \delta_{y_i, r} - \mathbf{w}_r^T \mathbf{x}_i \geq 1 - \xi_i$$

$$\forall i = 1, \dots, N, \quad \mathbf{w}_{u_i^*}^T \mathbf{x}_i^* - \mathbf{w}_{v_i^*}^T \mathbf{x}_i^* \leq \xi_i^* + \epsilon_1 \quad (5)$$

$$\forall p, q \in \{1, \dots, k\}, \quad -l \leq \sum_{i=1}^n \mathbf{w}_p^T \mathbf{x}_i - \sum_{i=1}^n \mathbf{w}_q^T \mathbf{x}_i \leq l,$$

where $\delta_{y_i, r}$ is the indication function, which equals 1 if r equals y_i and otherwise 0. Constraint (4) is the large margin constraint on the target examples. Constraint (5) minimizes the margins on the Universum examples. l is a parameter that controls the cluster balance to avoid trivially “optimal” solutions, which may assign most of the examples to one cluster. It is clear that the proposed formulation tries to maximize the margins/confidence on the target examples, and minimize the margins/confidence on Universum examples simultaneously. By doing so, the maximum margin clustering procedure can be guided by the prior knowledge from the Universum examples.

Next we will show the number of variables involved in problem (3) can be reduced by n . In particular, Eq.(3) is equivalent to the following optimization problem:

$$\min_{\mathbf{w}_p, \xi_i \geq 0, \xi_i^* \geq 0} \quad \frac{1}{2} \sum_{p=1}^k \|\mathbf{w}_p\|^2 + \frac{C_l}{n} \sum_{i=1}^n \xi_i + \frac{C_u}{N} \sum_{i=1}^N \xi_i^* \quad (6)$$

$$s.t. \quad \forall i = 1, \dots, n, \forall r = 1, \dots, k \quad (7)$$

$$\mathbf{w}_{u_i}^T \mathbf{x}_i + \delta_{u_i, r} - \mathbf{w}_r^T \mathbf{x}_i \geq 1 - \xi_i$$

$$\forall i = 1, \dots, N, \quad \mathbf{w}_{u_i^*}^T \mathbf{x}_i^* - \mathbf{w}_{v_i^*}^T \mathbf{x}_i^* \leq \xi_i^* + \epsilon_1 \quad (8)$$

$$\forall p, q \in \{1, \dots, k\}, \quad -l \leq \sum_{i=1}^n \mathbf{w}_p^T \mathbf{x}_i - \sum_{i=1}^n \mathbf{w}_q^T \mathbf{x}_i \leq l,$$

where $u_i = \arg \max_p \mathbf{w}_p^T \mathbf{x}_i$, and the optimal y_i equals u_i . Furthermore, for the convenience of computation, three *concatenated* vectors are introduced:

$$\tilde{\mathbf{w}} = [\mathbf{w}_1^T, \mathbf{w}_2^T, \dots, \mathbf{w}_p^T, \dots, \mathbf{w}_k^T]^T, \quad (9)$$

$$\mathbf{x}_{i(p)} = [\mathbf{0}, \dots, \mathbf{x}_i^T, \dots, \mathbf{0}]^T, \quad \mathbf{x}_{i(p)}^* = [\mathbf{0}, \dots, \mathbf{x}_i^{*T}, \dots, \mathbf{0}]^T,$$

where $\mathbf{0}$ is a $1 \times d$ zero vector. In $\mathbf{x}_{i(p)}$, only the $(p-1)d$ to pd -th elements are nonzero and equals \mathbf{x}_i and it is clear that $\tilde{\mathbf{w}}^T \mathbf{x}_{i(p)} = \mathbf{w}_p^T \mathbf{x}_i$. $\mathbf{x}_{i(p)}^*$ is defined similarly as $\mathbf{x}_{i(p)}$. We propose to absorb $\delta_{u_i, r}$ into ξ_i and use a separate variable ξ_{ir} for each constraint. The function $\mathbf{w}_{v_i^*}^T \mathbf{x}_i^*$ is non-convex. To simplify the formulation, we relax this function by $mean_{t \setminus u_i^*} \{\tilde{\mathbf{w}}^T \mathbf{x}_{i(t)}^*\}$. Here, the “mean” function calculates the average value of the input function with respect to the subscript variable. Then, Eq.(6) is rewritten as:

$$\min_{\tilde{\mathbf{w}}, \xi_i^* \geq 0, \xi_i^* \geq 0} \quad \frac{1}{2} \|\tilde{\mathbf{w}}\|^2 + \frac{C_l}{nk} \sum_{i=1}^n \sum_{r=1}^k \xi_{ir} + \frac{C_u}{N} \sum_{i=1}^N \xi_i^* \quad (10)$$

$$s.t. \quad \forall i = 1, \dots, n, \forall r = 1, \dots, k \quad (11)$$

$$\tilde{\mathbf{w}}^T \mathbf{x}_{i(u_i)} - \tilde{\mathbf{w}}^T \mathbf{x}_{i(r)} \geq 1 - \xi_{ir}$$

$$\forall i = 1, \dots, N, \quad (12)$$

$$\frac{k}{k-1} (\tilde{\mathbf{w}}^T \mathbf{x}_{i(u_i^*)} - mean_t \{\tilde{\mathbf{w}}^T \mathbf{x}_{i(t)}^*\}) \leq \xi_i^* + \epsilon_1$$

$$\forall p, q \in \{1, 2, \dots, k\}, \quad -l \leq \tilde{\mathbf{w}}^T \sum_{i=1}^n (\mathbf{x}_{i(p)} - \mathbf{x}_{i(q)}) \leq l.$$

This formulation is reasonable, however, it is both nonconvex and nonsmooth. Therefore, it cannot be solved directly. We will show how to solve this problem by using CCCP in the following section.

4.2 CCCP Decomposition

The nonconvexity in problem (10) is caused by the constraint (11) and (12). However, it can be considered as the difference between two different convex functions³. Therefore the Constrained Concave-Convex Procedure (CCCP) can be employed to solve this problem.

Given an initial point $\tilde{\mathbf{w}}^0$, CCCP computes $\tilde{\mathbf{w}}^{(t+1)}$ from $\tilde{\mathbf{w}}^{(t)}$ iteratively by replacing $\tilde{\mathbf{w}}^T \mathbf{x}_{i(u_i)}$ and $\tilde{\mathbf{w}}^T \mathbf{x}_{i(u_i^*)}$ in problem (10) with their corresponding first order Taylor expansions at $\tilde{\mathbf{w}}^{(t)}$ and solves the resulting quadratic programming problem, until convergence. Since $\tilde{\mathbf{w}}^T \mathbf{x}_{i(u_i)}$ and $\tilde{\mathbf{w}}^T \mathbf{x}_{i(u_i^*)}$ are non-smooth, their sub-gradients are used in their first order Taylor expansions. More precisely, for the t -th CCCP iteration, $\tilde{\mathbf{w}}^T \mathbf{x}_{i(u_i)}$ and $\tilde{\mathbf{w}}^T \mathbf{x}_{i(u_i^*)}$ are replaced by $\tilde{\mathbf{w}}^T \mathbf{x}_{i(u_i^{(t)})}$ and $\tilde{\mathbf{w}}^T \mathbf{x}_{i(u_i^{*(t)})}$, where $u_i^{(t)} = \arg \max_p (\tilde{\mathbf{w}}^{(t)})^T \mathbf{x}_{i(p)}$ and $u_i^{*(t)} = \arg \max_p (\tilde{\mathbf{w}}^{(t)})^T \mathbf{x}_{i(p)}^*$. In this way, problem (10) is decomposed into a set of convex subproblems and for the t -th CCCP iteration, the corresponding subproblem is:

$$\begin{aligned} \min_{\tilde{\mathbf{w}}, \xi_{ir} \geq 0, \xi_i^* \geq 0} \quad & \frac{1}{2} \|\tilde{\mathbf{w}}\|^2 + \frac{C_l}{nk} \sum_{i=1}^n \sum_{r=1}^k \xi_{ir} + \frac{C_u}{N} \sum_{i=1}^N \xi_i^* \quad (13) \\ \text{s.t.} \quad & \forall i = 1, \dots, n, \forall r = 1, \dots, k \\ & \tilde{\mathbf{w}}^T \mathbf{x}_{i(u_i^{(t)})} - \tilde{\mathbf{w}}^T \mathbf{x}_{i(r)} \geq 1 - \xi_{ir} \\ & \forall i = 1, \dots, N, \\ & \frac{k}{k-1} (\tilde{\mathbf{w}}^T \mathbf{x}_{i(u_i^{*(t)})}^* - \text{mean}_b \{\tilde{\mathbf{w}}^T \mathbf{x}_{i(b)}^*\}) \leq \xi_i^* + \epsilon_1 \\ & \forall p, q \in \{1, 2, \dots, k\}, \quad -l \leq \tilde{\mathbf{w}}^T \sum_{i=1}^n (\mathbf{x}_{i(p)} - \mathbf{x}_{i(q)}) \leq l. \end{aligned}$$

It is clear that by using CCCP together with the subgradient, during each iteration, we are trying to maximize the clustering margin on the target examples, and minimize the margin on the Universum examples, based on the cluster assignments obtained in the previous iteration. It has been verified that CCCP decreases the objective function monotonically [26], and is guaranteed to find a local optimal solution in a relatively small number of iterations. The whole algorithm is then described in Table 1. Here, $J^{(t)} = \frac{1}{2} \|\tilde{\mathbf{w}}^{(t)}\|^2 + \frac{C_l}{nk} \sum_{i=1}^n \sum_{r=1}^k \xi_{ir}^{(t)} + \frac{C_u}{N} \sum_{i=1}^N \xi_i^{*(t)}$.

The proposed algorithm is straightforward. But if the number of target examples, as well as the Universum examples goes large, or/and if the data dimension is high, directly solving problem (13) in the primal form may be very time consuming, since it involves a lot of optimization variables and constraints. There are many optimization alternatives to solve this efficiency problem [10, 24]. In this paper, we

³For constraint (11), $\tilde{\mathbf{w}}^T \mathbf{x}_{i(u_i)}$ is a convex and nonsmooth function, and $\tilde{\mathbf{w}}^T \mathbf{x}_{i(r)} - \xi_{ir}$ is convex and smooth. This constraint can be considered as the difference of these two convex functions, i.e., $(\tilde{\mathbf{w}}^T \mathbf{x}_{i(r)} - \xi_{ir}) - \tilde{\mathbf{w}}^T \mathbf{x}_{i(u_i)} \leq -1$. In the same way, constraint (12) can also be considered as the difference of two convex functions.

⁴The superscript t is used to denote that the result is obtained from the t -th CCCP iteration.

Algorithm: M²DCU

Input:

1. A set of target documents: $\mathcal{X} = \{\mathbf{x}_1, \dots, \mathbf{x}_n\}$
2. A set of Universum examples: $\mathcal{X}^* = \{\mathbf{x}_1^*, \dots, \mathbf{x}_N^*\}$
2. parameters: regularization parameter C_l and C_u , tolerance parameter for Universum examples $\epsilon_1 = 0.5$, CCCP solution precision $\epsilon_2 = 0.01$, cluster number k , cluster size balance parameter l

Output:

The cluster assignment \mathbf{y}

CCCP Iterations:

1. Construct $\tilde{\mathcal{X}} = \{\mathbf{x}_{i(r)}\}$, $\tilde{\mathcal{X}}^* = \{\mathbf{x}_{i(r)}^*\}$
2. Initialize $\tilde{\mathbf{w}}^0$, $t=0$, $\Delta J = 10^3$, $J^{(-1)} = 10^3$
3. **while** $\Delta J / J^{(t-1)} > \epsilon_2$ **do**
4. Derive problem (13) by updating $u_i^{(t)} = \arg \max_p (\tilde{\mathbf{w}}^{(t)})^T \mathbf{x}_{i(p)}$ and $u_i^{*(t)} = \arg \max_p (\tilde{\mathbf{w}}^{(t)})^T \mathbf{x}_{i(p)}^*$.
5. $t = t + 1$
6. Get $(\tilde{\mathbf{w}}^{(t)}, J^{(t)})$ by solving problem (13).
7. $\Delta J = J^{(t-1)} - J^{(t)}$
8. **end while**
9. **Cluster Assignment:**
For \mathbf{x}_i , the corresponding cluster assignment $\mathbf{y}_i = \arg \max_p (\tilde{\mathbf{w}}^{(t)})^T \mathbf{x}_{i(p)}$.

Table 1: Algorithm description: M²DCU

suggest a novel optimization method, which is an extension of the cutting plane method [10], to solve problem (13). The detail of this optimization method is elaborated in the Appendix A.

5. EXPERIMENTS

An extensive set of experiments is conducted to validate the effectiveness and the efficiency of the proposed method⁵. These experiments are conducted with Matlab r2008a on a 2.0GHZ Intel Quad Core computer with 4.0GB main memory.

5.1 Universum Examples

In practice, the Universum examples can be obtained easily. In this paper, three different ways are first employed to generate the Universum examples. A Universum selection method is then used to choose the most useful Universum examples.

$\mathcal{U}_{\text{rest}}$: some documents that are not included in the clustering tasks [28, 35]. We will elaborate $\mathcal{U}_{\text{rest}}$ for different clustering tasks when introducing datasets. Although this generation method sounds like needing category information, this is not necessary for a real application of clustering because it can obtain a separate set of documents that are not in categories in consideration. For example, we may be interested in clustering publications in some major computer science conferences (documents in major categories in their datasets), and we can assume $\mathcal{U}_{\text{rest}}$ consists of articles in a set of very specialized conferences on minority subject areas in computer science or even articles in another discipline.

$\mathcal{U}_{\text{rand}}$: Generate uniformly distributed features within the

⁵The code and data can be found on the author's homepage.

range of the min and max values on each dimension of the target examples [28, 35].

$\mathcal{U}_{\text{mean}}$: In [28], the authors designed some Universum examples which are combinations of different pictures from different categories. However, in a clustering problem, no label information is given. Therefore, an alternative method is used to design the Universum examples in the clustering problem. KMeans is first employed to pre-partition the whole dataset into k groups. $\frac{k(k-1)}{2}$ Universum examples are then generated by picking two clustering centers each time from the k groups and merging them together.

$\mathcal{U}_{\text{select}}$: In [4], the authors pointed out that the Universum examples that lie between different categories are the most helpful ones. Their basic motivation is reasonable. However, their method is only designed for classification problems, and is very time consuming. In this paper, we pick the most useful Universum examples for clustering in a different way. Our basic intuition is that the Universum examples that are closest to the target examples are the most useful ones for clustering. Although these Universum examples are close to the target examples, they do not belong to any of the clusters. So, they should have much higher impacts on the clustering results than the ones that lie far away. In order to measure the closeness of the Universum examples, a Gaussian Mixture Model is first trained on the target examples, with k being the number of gaussian models. Then, for each Universum example, we calculate the probability that it is generated by using this Gaussian Mixture Model. The Universum examples with the highest generation probabilities are then selected for use, and are referred to as $\mathcal{U}_{\text{select}}$. In our experiments, the top 10 percents of all the Universum examples are selected. As observed from these experiments, the selected proportions of $\mathcal{U}_{\text{rest}}$ and $\mathcal{U}_{\text{mean}}$ are much higher than that of $\mathcal{U}_{\text{rand}}$. This is because the randomly generated Universum often has a very low probability to be close to the target examples. But for $\mathcal{U}_{\text{rest}}$ and $\mathcal{U}_{\text{mean}}$, they are often closer to the target examples.

5.2 Datasets

We use four text datasets in our experiments.

20Newsgroup: This is a benchmark dataset⁶, which contains four main categories, i.e., ‘comp’, ‘rec’, ‘sci’, ‘talk’, as well as some other categories with small number of examples, such as ‘alt.atheism’, ‘misc.forsale’, etc. The four main categories are used for clustering, while examples in the remaining categories are used as $\mathcal{U}_{\text{rest}}$.

WebKB This dataset contains webpages from computer science departments at around four different universities⁷. They are divided into 7 categories (i.e., student, faculty, staff, course, project, department and other). We choose webpages from four categories as target examples for clustering (i.e., course, faculty, project, student), and the remaining 2 categories, i.e., “other” and “staff” will serve as $\mathcal{U}_{\text{rest}}$.

Reuters-Volume I (ReutersV1): It is an archive of over 800,000 manually categorized newswire stories [12]. A subset of ReutersV1 is used. There are in total 126 topics in it. In the experiments, we choose examples from three categories “ECAT”, “C151”, and “M14” in this subset as target examples, and some examples from the remaining categories

are randomly selected as $\mathcal{U}_{\text{rest}}$. To avoid some overlap problems, the target examples with more than one label from these three categories are removed.

Reuters21578: This dataset contains documents collected from the Reuters newswire in 1987⁸. There are in total 135 categories in this dataset. In our experiments, we use two subsets of this data collection, i.e., Reuters21578-1 and Reuters21578-2. Reuters21578-1 contains 6 categories among these 135 topics. Similar to ReutersV1, the documents associated with more than one of these 6 categories are not used. Reuters21578-2 contains another 10 categories. And the documents associated with more than one of these 10 categories are not used. For both of these sub-datasets, $\mathcal{U}_{\text{rest}}$ are randomly selected from examples in the remaining categories.

The basic information of these datasets is summarized in Table 2. For all of these documents, the tf-idf (normalized term frequency and log inverse document frequency) [15] features are extracted, with the stop words being removed and porter [15] as the stemmer. Furthermore, the most frequently appeared words are picked.

5.3 Methods

In experiments, the number of clusters k is set to be the true number of classes in target examples for all clustering algorithms. As described in Section 5.1, three different Universum methods are used in this paper. But we will not use them directly. The selection method mentioned previously is used to select the most useful Universum examples. $\mathcal{U}_{\text{select}}$ is then used for M²DCU as selected Universum examples. The class imbalance parameter l is set by grid search from the grid [0.001, 0.01, 0.1, 1, 10]. The parameter C_l is searched from the exponential grid $2^{[1:1:6]}$ and the parameter C_u is searched through [0.001, 0.01, 0.1, 1, 10]. To measure the effect brought by the Universum examples, we further report the performance of M²DCU-0, in which the C_u in problem (13) is set to be 0, and tune the other parameters using exactly the same way as we do for M²DCU.

Several different kinds of clustering methods are used for comparison. It includes: the probability generative method – Gaussian Mixture Model (GMM) [14]; the methods based on matrix factorizations, such as Nonnegative Matrix Factorization (NMF) [31] and Probabilistic Latent Semantic Analysis (PLSA) [8]; spectral/graph based methods, such as Normalized Cut (Ncut) [33], Clustering with Local and Global Consistency (CLGR) [27] and KMeans; Maximum Margin based clustering, such as CPM3C [36]. For both KMeans and GMM, for each experiment, the result is summarized over 50 independent runs to avoid local optimal. For Ncut, its gaussian similarity is determined by local scaling [34]. The formulation of CPM3C [36] is similar to M²DCU-0, but there is a significant difference in their optimization methods. The outer iteration of CPM3C is the cutting plane method and its inner iteration is CCCP. However, when using the cutting plane method to solve a nonconvex problem, the convergence property and optimality of the solution cannot be guaranteed. The parameters of CPM3C are tuned using exactly the same settings as M²DCU-0.

⁶<http://people.csail.mit.edu/jrennie/20Newsgroups/>

⁷<http://www.cs.cmu.edu/~webkb/>

⁸<http://www.daviddlewis.com/resources/testcollections/reuters21578/>

5.4 Evaluation Metrics

5.4.1 Clustering Accuracy (Acc)

The clustering accuracy (Acc) [25, 30, 36] is used to evaluate the final clustering performance. Specifically, we first take a set of labeled examples, remove the labels of these examples and run the clustering algorithms, then we relabel these examples using the clustering assignments returned by the algorithms. Finally, we measure the percentage of correct classifications by comparing the true labels and the labels assigned by the clustering algorithms as follows:

$$Acc = \frac{1}{n} \sum_{i=1}^n \delta(y_i, \text{map}(c_i)),$$

where, $\text{map}(\cdot)$ is a function that maps each cluster index to a class label, which can be found by the Hungarian algorithm [16]. c_i and y_i are the cluster index of \mathbf{x}_i and the true class label. $\delta(a, b)$ is a function that equals 1 when a equals b , and 0 otherwise.

5.4.2 Normalized Mutual Information (NMI)

Another evaluation metric we employed here is the Normalized Mutual Information (NMI) [22], which was originally a symmetric measure to quantify the statistical information shared between two distributions. More precisely, for two random variables X and Y , NMI is defined as:

$$NMI(X, Y) = \frac{I(X, Y)}{\sqrt{H(X)H(Y)}}. \quad (14)$$

Here, $I(X, Y)$ is mutual information between X and Y , while $H(X)$ and $H(Y)$ are entropies of X and Y . If there is a one to one correspondence between the distribution of X and that of Y , NMI equals one. From the perspective of information theory, it means all of the information encoded in X (sender) has already been correctly delivered to Y (receiver). Otherwise, if Y is merely a uniform distribution, NMI equals zero, which means no information would be transferred from X to Y . It is clear that $NMI(X, X) = 1$, which achieves the maximal possible value of NMI. Given two clustering results, we can consider the clustering assignments as distributions of random variables, and NMI in Eq.(14) can be estimated as:

$$NMI = \frac{\sum_{p=1}^k \sum_{q=1}^k n_{p,q} \log\left(\frac{n_{p,q}}{n_p \hat{n}_q}\right)}{\sqrt{(\sum_{p=1}^k n_p \log \frac{n_p}{n})(\sum_{q=1}^k \hat{n}_q \log \frac{\hat{n}_q}{n})}}, \quad (15)$$

where n_p refers to the number of data contained in the p -th cluster, \hat{n}_q is the number of data belonging to the q -th cluster, and $n_{p,q}$ denotes the number of data that are in the intersection between the p -th and the q -th clusters.

5.4.3 Rand Index (RandInd)

Suppose two clustering assignments, \mathcal{I} and \mathcal{L} , are given. $\mathcal{I} = \{\mathcal{I}_1, \mathcal{I}_2, \dots, \mathcal{I}_k\}$ represents the set of final clustering results and \mathcal{I}_i is the i -th cluster. $\mathcal{L} = \{\mathcal{L}_1, \mathcal{L}_2, \dots, \mathcal{L}_k\}$ denotes the set of true data classes such that \mathcal{L}_i represents the i -th class.

The rand index measures the similarity between \mathcal{I} and \mathcal{L} as [17]: $R = \frac{a+b}{a+b+c+d}$, where a is the number of data pairs in \mathcal{X} that are in both of these two clusters. b represents the number of data pairs in \mathcal{X} that are in different sets in these two clusters. c is the number of data pairs that are in the

same set in \mathcal{I} but in the different sets in \mathcal{L} . d denotes the number of data pairs in \mathcal{X} such that they are in different sets in \mathcal{I} but the same set in \mathcal{L} . It is natural to consider $a+b$ as the number of agreements between \mathcal{I} and \mathcal{L} , while $c+d$ represents the number of disagreements. If R equals 0, it means \mathcal{I} and \mathcal{L} are totally different, and if R equals 1, it means these two cluster assignments are exactly the same.

5.5 Clustering Results

The clustering results are reported from Table 3 to Table 5. In Table 6, the CPU running time of these different methods is also given.

From the clustering results, we can see that in all cases M²DCU performs better than M²DCU-0, which clearly demonstrates the benefits brought by the involvement of Universum examples. When compared with the probability generative methods, such as GMM, M²DCU prevails in all cases. For the methods based on matrix factorizations, such as NMF and PLSA, their performances are good in some cases. As can be seen from Table 4, the NMI performance of M²DCU on 20Newsgroup is worse than that of PLSA and NMF, and on Reuters21578-2, the Rand Index of NMF is also higher than M²DCU. But it is clear that for the other clustering measurements on these datasets, the proposed method performs better than NMF and PLSA. For example, for the clustering accuracy and Rand Index measurements on 20Newsgroup, the performances of the proposed method are higher than NMF and PLSA. It is safe to say that in most cases, the proposed method performs much better than the matrix factorization based methods. Ncut, CLGR, and KMeans are spectral based methods, whose performances mainly rely on graph factorization. These methods are reasonable, which place the cluster boundaries in the low density regions of the dataset. However, they do not utilize the Universum examples to further guide the clustering process, and therefore, their performances are inferior to that of M²DCU.

As a maximum margin clustering method, although the formulations of CPM3C and M²DCU-0 are similar, CPM3C uses the cutting plane method to solve a non-convex optimization problem. As previously claimed, the cutting plane method is not originally designed to solve non-convex optimization problems, and therefore, the performance of CPM3C cannot be guaranteed. Different from CPM3C, M²DCU uses the CCCP to solve the non-convex optimization problem, and the cutting plane method is suggested to solve the convex sub-problems derived from each CCCP iteration, and therefore it is theoretically much more elegant than CPM3C. As we shall see from experiments, in almost all cases, both M²DCU and M²DCU-0 perform better than CPM3C, which clearly validates our claims.

As for the CPU time, M²DCU is ranked among the top three fastest algorithms in three of the five datasets. For the experiments on Reuters21578-1 and Reuters21578-2, the proposed method are seemingly slower than some other algorithms. However, it is clear that the corresponding clustering performances of the proposed method are much higher than those of the ones with faster CPU running times. We suspect that the lower efficiency of the proposed method on these two datasets is caused by the data distributions in these two datasets, which increases the number of convergence steps. It is interesting to notice that on WebKB dataset the CPU running time of M²DCU is shorter than

| Dataset | Categories | #Dim | # PCA Dim | #Documents | # \mathcal{U}_{rest} | # \mathcal{U}_{rand} | # \mathcal{U}_{mean} | # $\mathcal{U}_{selected}$ |
|----------------|------------|------|-----------|------------|------------------------|------------------------|------------------------|----------------------------|
| 20Newsgroup | 4 | 6600 | 1025 | 16060 | 2768 | 1000 | 6 | 378 |
| WebKB | 5 | 4560 | 700 | 3223 | 3660 | 1000 | 10 | 467 |
| ReutersV1 | 3 | 8625 | 473 | 19619 | 7592 | 1000 | 3 | 860 |
| Reuters21578-1 | 6 | 927 | 189 | 8050 | 2192 | 1000 | 15 | 321 |
| Reuters21578-2 | 10 | 927 | 189 | 1689 | 4233 | 1000 | 45 | 528 |

Table 2: The detailed description of the datasets. # Dim represents the original data dimension, while #PCA Dim means the dimension after using Principal Component Analysis (PCA). For M²DCU, KMeans, Ncut, GMM, CPM3C, CLGR, the datasets are processed by PCA to increase the efficiency and reduce some noise. But for NMF, PLSA, since they require that the features should be non-negative, the dataset processed by PCA cannot be used. So, we run these two algorithms in the original feature space. The Universum examples used in experiments are denoted as $\mathcal{U}_{selected}$, which are selected from \mathcal{U}_{rest} , \mathcal{U}_{rand} and \mathcal{U}_{mean} .

| | M ² DCU | M ² DCU-0 | KMeans | NMF | Ncut | GMM | CPM3C | CLGR | PLSA |
|----------------|--------------------|----------------------|--------|------|------|------|-------|------|------|
| 20Newsgroup | 69.0 | 63.1 | 60.7 | 64.3 | 54.9 | 51.9 | 56.6 | 45.8 | 66.5 |
| WebKB | 62.6 | 59.2 | 48.3 | 47.8 | 47.2 | 60.4 | 51.4 | 48.9 | 52.7 |
| ReutersV1 | 95.7 | 94.8 | 61.1 | 84.4 | 75.2 | 69.1 | 88.1 | 80.4 | 85.6 |
| Reuters21578-1 | 76.6 | 73.2 | 45.6 | 57.4 | 62.6 | 59.4 | 60.6 | 47.6 | 48.7 |
| Reuters21578-2 | 53.2 | 50.7 | 38.4 | 50.0 | 40.4 | 41.8 | 43.9 | 38.7 | 47.1 |

Table 3: Clustering Accuracy (%) comparisons. The results marked with black represent the best performances on the corresponding datasets. M²DCU-0 represents the case when the weight of the Universum term C_u in Eq.(13) is set to be 0. It is clear that the Universum examples increase the maximum margin clustering performances dramatically. Although the formulations are similar, CPM3C and M²DCU-0 differ in their optimization methods. For KMeans and GMM, the results are summarized over 50 independent runs.

| | M ² DCU | M ² DCU-0 | KMeans | NMF | Ncut | GMM | CPM3C | CLGR | PLSA |
|----------------|--------------------|----------------------|--------|-------|-------|-------|-------|-------|--------------|
| 20Newsgroup | 0.408 | 0.361 | 0.357 | 0.443 | 0.252 | 0.205 | 0.278 | 0.209 | 0.476 |
| WebKB | 0.372 | 0.340 | 0.283 | 0.342 | 0.181 | 0.333 | 0.307 | 0.242 | 0.335 |
| ReutersV1 | 0.827 | 0.803 | 0.435 | 0.576 | 0.585 | 0.468 | 0.624 | 0.617 | 0.592 |
| Reuters21578-1 | 0.575 | 0.536 | 0.442 | 0.517 | 0.448 | 0.508 | 0.412 | 0.275 | 0.497 |
| Reuters21578-2 | 0.503 | 0.486 | 0.397 | 0.456 | 0.422 | 0.365 | 0.418 | 0.399 | 0.434 |

Table 4: Normalized Mutual Information (NMI) comparisons. The results marked with black represent the best performances on the corresponding datasets. M²DCU-0 represents the case when the weight of the Universum term C_u in Eq.(13) is set to be 0. Although in 20Newsgroup, the NMI of M²DCU is smaller than that of PLSA and NMF. But we can see that it improves the clustering results of M²DCU-0 with the use of Universum examples. And as we shall see, on the other two corresponding clustering indices, i.e, clustering accuracy, and rand index, M²DCU performs better than PLSA. For KMeans and GMM, the results are summarized over 50 independent runs.

| | M ² DCU | M ² DCU-0 | KMeans | NMF | Ncut | GMM | CPM3C | CLGR | PLSA |
|----------------|--------------------|----------------------|--------|--------------|-------|-------|-------|-------|-------|
| 20Newsgroup | 0.802 | 0.740 | 0.701 | 0.727 | 0.591 | 0.669 | 0.711 | 0.598 | 0.785 |
| WebKB | 0.794 | 0.698 | 0.654 | 0.740 | 0.643 | 0.689 | 0.667 | 0.609 | 0.737 |
| ReutersV1 | 0.952 | 0.934 | 0.618 | 0.814 | 0.778 | 0.715 | 0.855 | 0.839 | 0.825 |
| Reuters21578-1 | 0.840 | 0.799 | 0.741 | 0.778 | 0.673 | 0.784 | 0.691 | 0.614 | 0.759 |
| Reuters21578-2 | 0.854 | 0.808 | 0.800 | 0.860 | 0.792 | 0.834 | 0.748 | 0.813 | 0.847 |

Table 5: Rand Index comparisons. The results marked with black represent the best performances on the corresponding datasets. M²DCU-0 represents the case when the weight of the Universum term C_u in Eq.(13) is set to be 0. It is clear that M²DCU shows the best performances on most of these datasets. For KMeans and GMM, the results are summarized over 50 independent runs.

| | M ² DCU | M ² DCU-0 | KMeans | NMF | Ncut | GMM | CPM3C | CLGR | PLSA |
|----------------|--------------------|----------------------|--------|-------|-----------|--------|-------------|------------|--------------|
| 20Newsgroup | 801 | 789 | 15204 | 35515 | 151760 | 105440 | 22838 | 207752 | 10032 |
| WebKB | 232 | 307 | 1940 | 3465 | 771 | 18373 | 190 | 821 | 682 |
| ReutersV1 | 576 | 304 | 6303 | 32355 | 128810 | 21495 | 2264 | 162374 | 7551 |
| Reuters21578-1 | 685 | 662 | 3449 | 4065 | 1094 | 11445 | 382 | 4203 | 447 |
| Reuters21578-2 | 735 | 691 | 465 | 346 | 99 | 1550 | 256 | 119 | 197 |

Table 6: CPU Running Time (in seconds). For each dataset, the top 3 fastest algorithms are marked in black. M²DCU-0 represents the case when the weight of the Universum term C_u in Eq.(13) is set to be 0. From this table, it is clear that the efficiency of M²DCU is comparable with that of the other methods. Please note that the CPU running time for KMeans and GMM reported here are the results of 50 independent runs. It is interesting to note that in WebKB, M²DCU is faster than M²DCU-0, which indicates introducing the Universum examples may sometimes reduce the number of CCCP iterations.

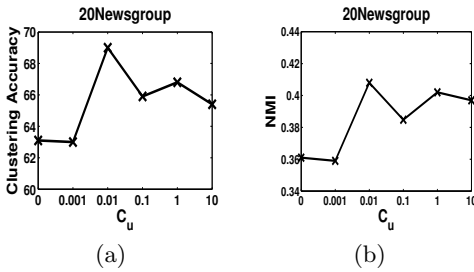


Figure 2: Parameter Sensibility on 20Newsgroup, with different C_u . It is clear that the performance is improved by incorporating the Universum examples.

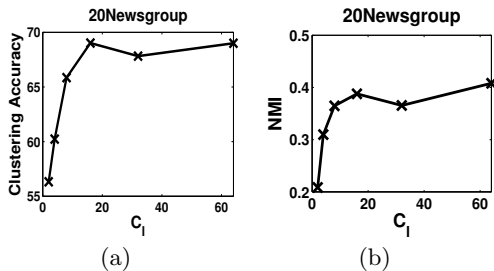


Figure 3: Parameter Sensibility on 20Newsgroup, with different C_l . From these two pictures, we can see that C_l is relatively stable in a very large range of the whole parameter space.

that of M^2DCU-0 . We suspect that sometimes, introducing the Universum examples can reduce the number of CCCP iterations and therefore result in a shorter running time.

Besides these comparison experiments, we also measure the parameter sensibility of the proposed method for C_u , C_l , l . The experiments are conducted on 20Newsgroup. In these experiments, C_u is tuned and evaluated through the grid $[0.001, 0.01, 0.1, 1, 10]$. C_l is from the grid $2^{[1:1:6]}$. And l is from the grid $[0.001, 0.01, 0.1, 1, 10]$. The parameter sensitivity experiments are shown from Fig.2 to Fig.4. In Fig.2, the performances with different C_u values are evaluated. C_u reflects the involvement of the Universum examples, where $C_u = 0$ means no involvement for the Universum examples. As can be seen from the trend of the performances, we can conclude that M^2DCU benefits a lot by introducing the Universum examples. In Fig.3, it is shown that the performance of the proposed method increases with C_l at first, and is relatively stable within a very large range of the whole parameter space, especially after C_l reaches a relatively large value. And in Fig.4, we can see that the proposed method is relatively stable with respect to l . But the performance of the proposed method does decrease a little after a fixed l value. Therefore, a relatively small value of l is normally suggested. We have also observed similar behaviors on the other three datasets, i.e., WebKB, ReutersV1 and Reuters21578. But due to the limit of space, we cannot put these experimental results here.

6. CONCLUSIONS

Document clustering is an important research topic with

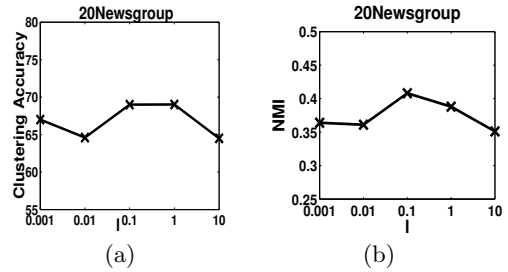


Figure 4: Parameter Sensibility on 20Newsgroup, with different l . The proposed method is relatively stable w.r.t l . But if l is too large, the performance will deteriorate. So, normally a relatively small value of l is suggested.

many practical applications. Previous document clustering methods only consider the examples that need to be clustered, but neglect the useful examples that can be obtained from some other sources. As a newly proposed concept, Universum, which is defined as a set of “non-examples” has been receiving more and more attentions. These examples can often be obtained easily and have already been shown the capability of improving the performances in classification problems. In this paper, we proposed a novel research problem – Document Clustering with Universum (DCU), which extends the idea of the utilization of Universum examples to document clustering problems, and suggest a method – Maximum Margin Document Clustering with Universum (M^2DCU) to solve this problem. The experimental results show that the Universum examples can be used to improve the performance of clustering under the framework of Maximum Margin Clustering, and the proposed method performs substantially better than state-of-the-art methods in most cases. In the future, we will further investigate: 1. how the Universum examples can be integrated to other clustering algorithms, such as KMeans, GMM, NMF; 2. how to design better ways to select Universum examples.

7. ACKNOWLEDGEMENT

We thank the anonymous reviewers for valuable comments. This research was partially supported by the NSF research grants IIS-0746830, CNS-1012208, IIS-1017837, CCF- 0939370.

References

- [1] R. Bekkerman, H. Raghavan, J. Allan, and K. Eguchi. Interactive clustering of text collections according to a user-specified criterion. In *IJCAI*, pages 684–689, 2007.
- [2] R. Bekkerman, S. Zilberstein, and J. Allan. Web page clustering using heuristic search in the web graph. In *IJCAI*, pages 2280–2285. IJCAI, 2006.
- [3] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, March 2004.
- [4] S. Chen and C. Zhang. Selecting informative universum sample for semi-supervised learning. In *IJCAI*, pages 1016–1021, 2009.
- [5] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification (2nd Edition)*. Wiley-Interscience, 2 edition, November 2001.
- [6] J. Gong and D. W. Oard. Selecting hierarchical clustering cut points for web person-name disambiguation. In *Proceedings of the 32nd international ACM SIGIR conference on*

Research and development in information retrieval, SIGIR '09, pages 778–779, New York, NY, USA, 2009. ACM.

- [7] J. He, E. J. Meij, and M. de Rijke. Result diversification based on query-specific cluster ranking. *Journal of the American Society of Information Science and Technology*, 2011. To appear.
- [8] T. Hofmann. Probabilistic latent semantic indexing. In *SIGIR*, pages 50–57, 1999.
- [9] K. Huang, Z. Xu, I. King, and M. R. Lyu. Semi-supervised learning from general unlabeled data. In *ICDM*, pages 273–282, 2008.
- [10] T. Joachims. Training linear svms in linear time. In *KDD*, pages 217–226, 2006.
- [11] J. Kelley Jr. The cutting-plane method for solving convex programs. *Journal of the SIAM*, pages 703–712, 1960.
- [12] D. D. Lewis, Y. Yang, T. G. Rose, and F. Li. Rcv1: A new benchmark collection for text categorization research. *Journal of Machine Learning Research*, 5:361–397, 2004.
- [13] Q. Li, B. M. Kim, and S.-H. Myaeng. Clustering for probabilistic model estimation for cf. In *WWW (Special interest tracks and posters)*, pages 1104–1105, 2005.
- [14] X. Liu, Y. Gong, W. Xu, and S. Zhu. Document clustering with cluster refinement and model selection capabilities. In *SIGIR*, pages 191–198, 2002.
- [15] C. D. Manning, P. Raghavan, and H. Schütze. *Introduction to Information Retrieval*. Cambridge University Press, 1 edition, July 2008.
- [16] C. H. Papadimitriou and K. Steiglitz. *Combinatorial Optimization : Algorithms and Complexity*. Dover Publications, July 1998.
- [17] W. M. Rand. Objective Criteria for the Evaluation of Clustering Methods. *Journal of the American Statistical Association*, 66(336):846–850, 1971.
- [18] N. Sahoo, J. Callan, R. Krishnan, G. T. Duncan, and R. Padman. Incremental hierarchical clustering of text documents. In *CIKM*, pages 357–366, 2006.
- [19] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 22(8):888–905, 2000.
- [20] F. H. Sinz, O. Chapelle, A. Agarwal, and B. Schölkopf. An analysis of inference with the universum. In *NIPS*, 2007.
- [21] M. Spitters and W. Kraaij. Unsupervised clustering in multilingual news streams. In *LREC 2002 workshop: Event Modelling for Multilingual Document Linking*, pages 42–46, 2002.
- [22] A. Strehl and J. Ghosh. Cluster ensembles — a knowledge reuse framework for combining multiple partitions. *Journal of Machine Learning Research*, 3:583–617, 2002.
- [23] T. Tao and C. Zhai. A mixture clustering model for pseudo feedback in information retrieval. In *IFCS*, 2004.
- [24] C. H. Teo, S. V. N. Vishwanathan, A. J. Smola, and Q. V. Le. Bundle methods for regularized risk minimization. *Journal of Machine Learning Research*, 11:311–365, 2010.
- [25] H. Valizadegan and R. Jin. Generalized maximum margin clustering and unsupervised kernel learning. In *NIPS*, pages 1417–1424, 2006.
- [26] A. S. Vishwanathan, A. J. Smola, and S. V. N. Vishwanathan. Kernel methods for missing variables. In *AI-STAT*, pages 325–332, 2005.
- [27] F. Wang, C. Zhang, and T. Li. Regularized clustering for documents. In *SIGIR*, pages 95–102, 2007.
- [28] J. Weston, R. Collobert, F. H. Sinz, L. Bottou, and V. Vapnik. Inference with the universum. In *ICML*, pages 1009–1016, 2006.
- [29] E. P. Xing, A. Y. Ng, M. I. Jordan, and S. J. Russell. Distance metric learning with application to clustering with side-information. In *NIPS*, pages 505–512, 2002.
- [30] L. Xu, J. Neufeld, B. Larson, and D. Schuurmans. Maximum margin clustering. In *NIPS*, 2004.
- [31] W. Xu, X. Liu, and Y. Gong. Document clustering based on non-negative matrix factorization. In *SIGIR*, pages 267–273, 2003.
- [32] H. Yang and J. P. Callan. Near-duplicate detection by

instance-level constrained clustering. In *SIGIR*, pages 421–428. ACM, 2006.

- [33] S. X. Yu and J. Shi. Multiclass spectral clustering. In *ICCV*, pages 313–319, 2003.
- [34] L. Zelnik-Manor and P. Perona. Self-tuning spectral clustering. In *NIPS*, 2004.
- [35] D. Zhang, J. Wang, F. Wang, and C. Zhang. Semi-supervised classification with universum. In *SDM*, pages 323–333, 2008.
- [36] B. Zhao, F. Wang, and C. Zhang. Efficient multiclass maximum margin clustering. In *ICML*, pages 1248–1255, 2008.

APPENDIX

A. EFFICIENT OPTIMIZATION

Formulation (13) is a standard quadratic programming subproblem for the t -th CCCP iteration. But if the number of constraints, i.e., the number of target examples and Universum examples, is huge, it would be very time consuming to solve this problem. Some of the previous large scale optimization methods can be adapted to solve this problem, such as [10] and [24]. In this section, inspired by [10], the cutting plane method [11], which has shown its effectiveness and efficiency in solving similar tasks, is suggested to solve this subproblem. But different from [10], which is mainly designed for optimization problems with only one set of constraints, in this paper, we propose a novel optimization method – Two-View Cutting Plane method that can be used to solve the optimization problem with two different sets of constrains, as is the case in problem (13).

Similar to [10], problem (13) is first transformed to the following equivalent form:

$$\begin{aligned} \min_{\tilde{\mathbf{w}}, \xi \geq 0, \xi^* \geq 0} \quad & \frac{1}{2} \|\tilde{\mathbf{w}}\|^2 + C_l \xi + C_u \xi^* \quad (16) \\ \text{s.t.} \quad & \forall \mathbf{c} \in \{0, 1\}^{n \times k} \\ & \tilde{\mathbf{w}}^T \frac{1}{nk} \sum_{i=1}^n \sum_{r=1}^k \mathbf{c}_{ir} (\mathbf{x}_{i(u_i^{(t)})} - \mathbf{x}_{i(r)}) \geq \frac{1}{nk} \sum_{i=1}^n \sum_{r=1}^k \mathbf{c}_{ir} - \xi \\ & \forall \mathbf{d} \in \{0, 1\}^N \\ & \frac{k}{N(k-1)} \sum_{i=1}^N \mathbf{d}_i (\tilde{\mathbf{w}}^T \mathbf{x}_{i(u_i^{(t)})}^* - \text{mean}_b \{\tilde{\mathbf{w}}^T \mathbf{x}_{i(b)}^*\}) \leq \xi^* + \frac{\epsilon_1}{N} \sum_{i=1}^N \mathbf{d}_i \\ & \forall p, q \in \{1, 2, \dots, k\}, \quad -l \leq \tilde{\mathbf{w}}^T \sum_{i=1}^n (\mathbf{x}_{i(p)} - \mathbf{x}_{i(q)}) \leq l \end{aligned}$$

ξ and ξ^* are two newly introduced variables, where $\xi = \frac{1}{nk} \sum_{i=1}^n \sum_{r=1}^k \xi_{ir}$ and $\xi^* = \frac{1}{N} \sum_{i=1}^N \xi_i^*$. By doing this, the number of variables that need to be optimized is further reduced by $(n \times k + N - 2)$. However, it contains a large number of constraints. Fortunately, we don't need to take all of these constraints into account. To solve the problem, the cutting plane algorithm is adapted to find two polynomially sized subset of constraints Ω and Ω^* from the whole set of constraints $\{0, 1\}^{n \times k}$ and $\{0, 1\}^N$ respectively in problem (16). In particular, the constraint sets Ω and Ω^* are initially set to be empty, and are built step by step until the solution of the relaxed problem satisfies all the constraints up to the two cutting plane precisions ϵ_3 and ϵ_4 , i.e., $\forall \mathbf{c} \in \{0, 1\}^{n \times k}$,

$$\tilde{\mathbf{w}}^T \frac{1}{nk} \sum_{i=1}^n \sum_{r=1}^k \mathbf{c}_{ir} (\mathbf{x}_{i(u_i^{(t)})} - \mathbf{x}_{i(r)}) \geq \frac{1}{nk} \sum_{i=1}^n \sum_{r=1}^k \mathbf{c}_{ir} - (\xi + \epsilon_3) \quad (17)$$

and $\forall \mathbf{d} \in \{0, 1\}^N$,

$$\begin{aligned} & \frac{k}{N(k-1)} \sum_{i=1}^N \mathbf{d}_i (\tilde{\mathbf{w}}^T \mathbf{x}_{i(u_i^*(t))}^* - \text{mean}_b \{ \tilde{\mathbf{w}}^T \mathbf{x}_{i(b)}^* \}) \\ & \leq \xi^* + \epsilon_4 + \frac{\epsilon_1}{N} \sum_{i=1}^N \mathbf{d}_i \end{aligned} \quad (18)$$

It means, the remaining exponential number of the two sets of constraints will not be violated up to the precision ϵ_3 and ϵ_4 respectively. Therefore, we don't need to explicitly add them to Ω and Ω^* .

The cutting plane algorithm starts with two empty sets of constraints Ω and Ω^* as follows:

$$\begin{aligned} \min_{\tilde{\mathbf{w}}, \xi \geq 0, \xi_i^* \geq 0} & \quad \frac{1}{2} \|\tilde{\mathbf{w}}\|^2 + C_l \xi + C_u \xi^* \\ \text{s.t.} & \quad \forall p, q \in \{1, 2, \dots, k\} \\ & \quad -l \leq \tilde{\mathbf{w}}^T \sum_{i=1}^n (\mathbf{x}_{i(p)} - \mathbf{x}_{i(q)}) \leq l \end{aligned} \quad (19)$$

After getting the solution $\tilde{\mathbf{w}}^{(t_0)}$ ⁹ of the above optimization problem, the most violated constraints can be computed as:

$$\mathbf{c}_{ir}^{t_0} = \begin{cases} 1, & \text{if } (\tilde{\mathbf{w}}^{t_0})^T \mathbf{x}_{i(u_i^{(t)})} - (\tilde{\mathbf{w}}^{t_0})^T \mathbf{x}_{i(r)} \leq 1 \\ 0, & \text{otherwise} \end{cases} \quad (20)$$

and

$$\mathbf{d}_i^{t_0} = \begin{cases} 1, & \text{if } (\tilde{\mathbf{w}}^{t_0})^T \mathbf{x}_{i(u_i^{*(t)})}^* - \text{mean}_b \{ (\tilde{\mathbf{w}}^{t_0})^T \mathbf{x}_{i(b)}^* \} \geq \epsilon_1 \\ 0, & \text{otherwise} \end{cases} \quad (21)$$

Then, \mathbf{c}^{t_0} will be added to Ω if H^{t_0} is true, and \mathbf{d}^{t_0} will be added to Ω^* if H^{*t_0} is true. Here, H^{t_s} is used to denote the condition $(\tilde{\mathbf{w}}^{(t_s)})^T \frac{1}{nk} \sum_{i,r} \mathbf{c}_{ir}^{t_s} (\mathbf{x}_{i(u_i^{(t)})} - \mathbf{x}_{i(r)}) \leq \frac{1}{nk} \sum_{i,r} \mathbf{c}_{ir}^{t_s} - (\xi^{(t_s)} + \epsilon_3)$, and H^{*t_s} indicates whether $\frac{1}{N(k-1)} \sum_{i=1}^N \mathbf{d}_i^{t_s} (\tilde{\mathbf{w}}^T \mathbf{x}_{i(u_i^{*(t)})}^* - \text{mean}_b \{ \tilde{\mathbf{w}}^T \mathbf{x}_{i(b)}^* \}) \geq \xi^{*(t_s)} + \epsilon_4 + \frac{\epsilon_1}{N} \sum_{i=1}^N \mathbf{d}_i^{t_s}$ holds or not. After updating Ω and Ω^* , the optimization problem becomes:

$$\begin{aligned} \min_{\tilde{\mathbf{w}}, \xi \geq 0, \xi_i^* \geq 0} & \quad \frac{1}{2} \|\tilde{\mathbf{w}}\|^2 + C_l \xi + C_u \xi^* \\ \text{s.t.} & \quad \forall \mathbf{c} \in \Omega, \\ & \quad \tilde{\mathbf{w}}^T \frac{1}{nk} \sum_{i=1}^n \sum_{r=1}^k \mathbf{c}_{ir} (\mathbf{x}_{i(u_i^{(t)})} - \mathbf{x}_{i(r)}) \geq \frac{1}{nk} \sum_{i=1}^n \sum_{r=1}^k \mathbf{c}_{ir} - \xi \\ & \quad \forall \mathbf{d} \in \Omega^*, \\ & \quad \frac{k}{N(k-1)} \sum_{i=1}^N \mathbf{d}_i (\tilde{\mathbf{w}}^T \mathbf{x}_{i(u_i^{*(t)})}^* - \text{mean}_b \{ \tilde{\mathbf{w}}^T \mathbf{x}_{i(b)}^* \}) \\ & \quad \leq \xi^* + \frac{\epsilon_1}{N} \sum_{i=1}^N \mathbf{d}_i \\ & \quad \forall p, q \in \{1, 2, \dots, k\}, \quad -l \leq \tilde{\mathbf{w}}^T \sum_{i=1}^n (\mathbf{x}_{i(p)} - \mathbf{x}_{i(q)}) \leq l \end{aligned} \quad (22)$$

⁹Throughout this paper, t_s is used to denote the s -th iteration of the cutting plane iteration for solving the problem derived from the t -th CCCP iteration.

| Algorithm: Two-View Cutting Plane |
|--|
| <p>Input:</p> <ol style="list-style-type: none"> 1. Problem (13). 2. Cutting Plane precision for View 1: $\epsilon_3 = 0.01$, for View 2: $\epsilon_4 = 0.01$ <p>Output:</p> <p>The cluster hyperplane $\tilde{\mathbf{w}}^{(t)}$ and the objective function value $J^{(t)}$</p> <ol style="list-style-type: none"> 1. $s = -1$. $\Omega = \emptyset$, $\Omega^* = \emptyset$, $H^{t_s} = \text{true}$, $H^{*t_s} = \text{true}$ 2. while $H^{t_s} H^{*t_s}$ is true do 3. $s = s + 1$ 4. Get $(\tilde{\mathbf{w}}^{(t_s)}, \xi^{(t_s)}, \xi^{*(t_s)})$ by solving (22) under Ω and Ω^*. 5. Compute the most violated constraints, i.e., $\mathbf{c}_{ir}^{t_s}$ and $\mathbf{d}_i^{t_s}$, by $\mathbf{c}_{ir}^{t_s} = \begin{cases} 1, & \text{if } (\tilde{\mathbf{w}}^{(t_s)})^T \mathbf{x}_{i(u_i^{(t)})} - (\tilde{\mathbf{w}}^{(t_s)})^T \mathbf{x}_{i(r)} \leq 1 \\ 0, & \text{otherwise} \end{cases}$ and $\mathbf{d}_i^{t_s} = \begin{cases} 1, & \text{if } (\tilde{\mathbf{w}}^{(t_s)})^T (\mathbf{x}_{i(u_i^{*(t)})}^* - \text{mean}_b \{ \mathbf{x}_{i(b)}^* \}) \geq \epsilon_1 \\ 0, & \text{otherwise} \end{cases}$ 6. Update H^{t_s} and H^{*t_s}. 7. If H^{t_s} is true, then update the constraint set Ω by $\Omega = \Omega \cup \mathbf{c}^{t_s}$; If H^{*t_s} is true, then update the constraint set Ω^* by $\Omega^* = \Omega^* \cup \mathbf{d}^{t_s}$ 8. end while 9. $\tilde{\mathbf{w}}^{(t)} = \tilde{\mathbf{w}}^{t_s}$ 10. $J^{(t)} = \frac{1}{2} \ \tilde{\mathbf{w}}^{(t)}\ ^2 + C_l \xi^{(t_s)} + C_u \xi^{*(t_s)}$ |

Table 7: Algorithm description: Two-View Cutting Plane. It can be used to solve the step 6 in Table 1 efficiently.

Both Ω and Ω^* contain at most one constraint vector at this time. From this updated optimization problem, we can get the solution $\tilde{\mathbf{w}}^{(t_1)}$. Then, the constraints \mathbf{c}^{t_1} and \mathbf{d}^{t_1} can be computed similarly as that in Eq.(20) and Eq.(21), where the only difference is that $\tilde{\mathbf{w}}^{(t_0)}$ would be replaced by $\tilde{\mathbf{w}}^{(t_1)}$. The procedure is repeated until all of the constraints satisfy the requirement in Eq.(17) and Eq.(18), i.e., both H^{t_s} and H^{*t_s} are false. In this way, a successive strengthening approximation series of the problem (13) can be constructed by the expanding number of cutting plane procedures that cut off the current optimal solution from the feasible set [11].

The concrete procedure is summarized in Table 7. In Step 4, the problem can also be solved efficiently through the dual form [3], since there are only a few constraints involved. Here, due to the limit of space, the concrete dual form is omitted.

It is clear that the Two-View Cutting Plane algorithm is an iterative method. One of the nice properties of employing this method is the guaranteed fast convergence rate as can be derived similar to that in [10]. So, the algorithm will converge with polynomial number of constraints in a linear time, and therefore is much faster than directly solving the optimization problem (13).