Intelligent Data Analysis 14 (2010) 479–495
DOI 10.3233/IDA-2010-0433
IOS Press

Clustering with feature order preferences

Jun Sun[a,d,*], Wenbo Zhao[b], Jiangwei Xue[c], Zhiyong Shen[a,d] and Yidong Shen[a]
[a]*State Key Laboratory of Computer Science, Institute of Software, Chinese Academy of Sciences, Beijing, China*
[b]*Department of Computer Science and Engineering, University of California, San Diego, La Jolla, CA, USA*
[c]*Department of Mathematics, The Pennsylvania State University, Pennsylvania, PA, USA*
[d]*Graduate University, Chinese Academy of Sciences, Beijing, China*

Abstract. We propose a clustering algorithm that effectively utilizes feature order preferences, which have the form that feature s is more important than feature t. Our clustering formulation aims to incorporate feature order preferences into prototype-based clustering. The derived algorithm automatically learns distortion measures parameterized by feature weights which will respect the feature order preferences as much as possible. Our method allows the use of a broad range of distortion measures such as Bregman divergences. Moreover, even when generalized entropy is used in the regularization term, the subproblem of learning the feature weights is still a convex programming problem. Empirical results on some datasets demonstrate the effectiveness and potential of our method.

Keywords: Clustering, domain knowledge, Bregman divergence, feature order preferences, entropy regularization, prototype-based clustering, convex optimization, quadratic programming

1. Introduction

Data clustering (a.k.a. cluster analysis) is a fundamental technique of unsupervised learning that has been extensively studied for several decades [13,14], and yet is still an active area in data mining and machine learning research. Given a set of objects, usually in the form of data points in $\mathbb{R}^d$, clustering aims to partition them into several disjoint groups (called clusters) in a meaningful and natural way. The clustering objective is often formulated to maximize intra-cluster cohesion and inter-cluster separability. Many clustering techniques have emerged over the years and have been widely applied to many different tasks, such as image segmentation [24], unsupervised document clustering [9,28], organizing online news articles, grouping genes and proteins with similar functionality, music clustering, customer and market segmentation, and so on.

In supervised learning, labeled data can be used to guide the learning procedure to obtain the most accurate model. However, in an unsupervised learning setting, no expert-provided labels exist for the clustering algorithm. Therefore, it is very difficult to define precisely which clustering of the data is the best one [10]. To make things worse, there may be multiple meaningful and natural clusterings in the same dataset. Thus, some clustering formulations aim to uncover disparate or alternative clusterings in

[*]Corresponding author: Jun Sun, State Key Laboratory of Computer Science, Institute of Software, Chinese Academy of Sciences; P.O. Box 8718, 4# South Fourth Street, Zhong Guan Cun, Beijing 100190, China.

a single dataset that provide multiple different views of the data [15]. However, the problem of selecting the best clustering still remains unsolved.

Since no explicit label information exists in the standard clustering setting, heavy assumptions about similarity or distance have to be made to measure the goodness of a clustering. In practice, false assumptions that're been made may lead to meaningless clusterings. However, when some extra domain knowledge is available, this problem will be alleviated. Then it becomes much easier to find a reasonable clustering for the task at hand. The problem of how to effectively incorporate domain knowledge into a clustering system has been an active topic in machine learning and data mining research in recent years. For example, the problem of clustering with instance-level knowledge in the form of pairwise constraints, namely, *must-link* and *cannot-link* constraints, has received a significant amount of attention in recent years. A lot of techniques [5,17,18,27] has been proposed about clustering with must-link and cannot-link constraints, which is also known as semi-supervised clustering.

Previous work on clustering with domain knowledge generally utilizes instance-level knowledge such as pairwise constraints. Instead, we consider feature-level domain knowledge. In particular, we consider a new form of domain knowledge called *feature order preferences*. Feature order preferences have the form that feature s is more important than feature t. Obviously, feature order preferences are much easier to obtain than precise relative weights of the features. This work is inspired by the research on how to utilize instance-level order preferences in ranking and regression problems [8,31].

In this paper, we propose a novel clustering algorithm which is able to take into account feature order preferences effectively. Our proposed clustering formulation aims to incorporate distance learning into prototype-based clustering, where the distortion measure is parameterized by the feature weights which will respect the feature order preferences as much as possible. Our clustering objective function allows the use of any Bregman divergence [3], which is a large family of distortion measures including squared Euclidean distance and Kullback-Leibler divergence. For the regularization term, we use a recently proposed definition of generalized entropy [19], which is very general that can lead to many instantiations of our clustering formulation. An important component in our algorithm is the subproblem of feature weight learning. Even when generalized entropy is used in the regularization term, this problem is still a convex programming problem, so efficient and effective algorithms exist [7]. We discuss ways to extend our clustering algorithm to accommodate other classes of distortion measures such as directional similarity functions (cosine similarity and Pearson's correlation) [5,16]. Besides, we also explain how to extend our clustering framework to deal with discrete data with nominal attributes, with some discussion about the relations to other previously proposed categorical clustering formulations. In our experimental section, the clustering results on several real-world datasets demonstrate the effectiveness and potential of our proposed clustering algorithm with feature order preferences.

The rest of the paper is organized as follows. In Section 2, we formulate the clustering objective function in detail. The proposed clustering algorithm is derived in Section 3. In Section 4, several extensions are explained and discussed. Then, we evaluate the proposed clustering method using several datasets in Section 5. We conclude the paper and discuss some future works in Section 6.

2. Model formulation

Let $\mathcal{F} \subseteq \mathbb{R}^d$ denote the input space from which n data points, $\mathbf{x}_1, \cdots, \mathbf{x}_n$, where $\mathbf{x}_i = [x_{i1}, \ldots, x_{id}]^\top$, are sampled. We use $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n]^\top \in \mathbb{R}^{n \times d}$ to denote the input data matrix.

Given k and $\{\mathbf{x}_i\}_{i=1}^n \subseteq \mathcal{F}$, the goal of clustering is to find a disjoint partitioning $\{\pi_c\}_{c=1}^k$ of the data where π_c is the c-th cluster. $n_c = |\pi_c|$ is the number of points in the c-th cluster.

We introduce some notational conventions first. Boldface lowercase letters, such as $\mathbf{x}$ and $\mathbf{y}$, denote column vectors. The superscript $\top$ is used to denote the transpose of a vector. $\mathbf{1}_d \in \mathbb{R}^d$ denotes the d-dimensional column vector whose entries are all 1's and $\mathbf{I}_d \in \mathbb{R}^{d\times d}$ denotes the identity matrix. We use $\log(\cdot)$ to denote natural logarithm. $\mathbb{R}_+$ and $\mathbb{R}_{++}$ denote the set of nonnegative and positive real numbers respectively. $\Delta_d = \left\{\mathbf{w} \in \mathbb{R}^d_+ \,|\, \mathbf{w}^\top \mathbf{1}_d = 1\right\}$ is the probability simplex. For any $\mathbf{w} = [w_1, \cdots, w_d]^\top \in \Delta_d$, the elements $\{w_j\}_{j=1}^d$ of $\mathbf{w}$ are nonnegative and sum to one.

2.1. *Clustering objective with feature order preferences*

In practice, each feature (or dimension) of the data points may be of different importance with respect to the current clustering task. We use $\mathbf{w} = [w_1, \ldots, w_d]^\top \in \Delta_d$ to denote a feature weight vector so that w_j $(1 \leqslant j \leqslant d)$ indicates the relative importance of feature j. For example, $\mathbf{w} = [\frac{1}{d}, \ldots, \frac{1}{d}]^\top$ is the uniform feature weighting where all the features are of equal importance.

In our clustering formulation, we assume that some domain knowledge in the form of *feature order preferences* will be given. A feature order preference is defined as a tuple (s, t, δ), which is interpreted as $w_s - w_t \geqslant \delta$. Thus, $\delta > 0$ means that feature s is more important than feature t. "Features s and t are of approximately equal importance" can be interpreted as $|w_s - w_t| \leqslant \epsilon$, hence can be encoded by a combination of $(s, t, -\epsilon)$ and $(t, s, -\epsilon)$ where ϵ is a small positive real number. In practice, optimal feature weights are often very difficult to obtain, but feature order preferences sometimes can be easy to acquire even when a domain expert has just some vague idea about the relative importance of the features. We assume that a set of m feature order preferences, denoted by $\mathcal{P} = \{(s_i, t_i, \delta_i)\}_{i=1}^m$ where $|\mathcal{P}| = m$, will be given.

A key component in a typical clustering formulation is the dissimilarity (or similarity) between two points measured by a distortion function. Traditionally, without any domain knowledge, each dimension of the data points are often assumed to contribute equally to the distortion measure. Now that a set of feature order preferences is given by domain experts, we want to learn a distortion measure parameterized by the feature weight vector $\mathbf{w} \in \Delta_d$. Furthermore, we want to incorporate the process of distortion measure learning into prototype-based clustering to produce more accurate clusterings.

Our clustering objective function consists of three terms which will be explained in detail as follows.

– First, we want to minimize the *intra-cluster distortion* of the clusters $\{\pi_c\}_{c=1}^k$. Assume that there's a cluster representative $\mu_c \in \mathcal{F}$ for each cluster π_c. The distortion of cluster π_c is measured by $\sum_{\mathbf{x}_i\in\pi_c} \mathsf{D}_{\mathbf{w}}(\mathbf{x}_i, \mu_c)$ where $\mathsf{D}_{\mathbf{w}}(\cdot,\cdot)$ is the distortion measure between two data points parameterized by the feature weight vector $\mathbf{w} \in \Delta_d$. The quality of the entire clustering $\{\pi_c\}_{c=1}^k$ is measured by the total distortion of all the k clusters, namely,

$$\sum_{c=1}^{k} \sum_{\mathbf{x}_i\in\pi_c} \mathsf{D}_{\mathbf{w}}(\mathbf{x}_i, \mu_c). \tag{1}$$

– Second, we want the weight vector $\mathbf{w}$ to respect the feature order preferences in $\mathcal{P}$ as much as possible. Note that we treat the preferences as soft constraints rather than hard ones. A *penalty term* will be added to the objective function so that more violations of the preferences will lead to larger penalties. Besides, we don't penalize those constraints that're not violated. Thus, if all the weights are consistent with all the preferences, then the penalty term will be zero. Therefore, we use a shifted hinge function [31] in the penalty term: for $p = (s, t, \delta) \in \mathcal{P}$, the penalty term for p is

$\max(\delta - (w_s - w_t), 0)$. Using m auxiliary variables $\xi = [\xi_p] \in \mathbb{R}^m$ where $p \in \mathcal{P}$, minimizing the penalty term is equivalent to the following optimization problem:

$$\begin{aligned} \min_{\mathbf{w},\,\xi} \quad & \sum_{p \in \mathcal{P}} \xi_p + \text{Other terms.} \\ \text{subjectto} \quad & \mathbf{w} \in \Delta_d \\ & w_s - w_t \geqslant \delta - \xi_p \quad \text{forall} p = (s,t,\delta) \in \mathcal{P} \\ & \xi_p \geqslant 0 \quad \text{forall} p \in \mathcal{P} \end{aligned} \tag{2}$$

– Third, aside from the feature order preferences, we don't want to make further unwarranted assumptions about the values of the weights. Therefore, another *regularization term*, $-\widehat{H}(\mathbf{w})$, is added to the objective function to ensure that the weights are as uniform as possible. If the regularization term is missing and no feature order preferences are provided, the feature with the least intra-cluster distortion will get all the weight (feature weight = 1), which is undesirable and should be avoided. $\widehat{H}(\cdot)$ is the generalized entropy which will be defined and discussed in Section 4.2. A key property of $\widehat{H}(\cdot)$ is that the more uniform the weights w_j $(1 \leqslant j \leqslant d)$ are, the larger the value of $\widehat{H}(\mathbf{w})$ becomes. For the algorithm derived in the next section, we'll use $\widehat{H}(\mathbf{w}) = 1 - \mathbf{w}^\top \mathbf{w}$ which will be referred to as ℓ_2-*entropy*. Extensions will be discussed in Section 4.2.

By combining the three terms discussed above, we have the overall clustering goal, which is to minimize the following clustering objective function:

$$\sum_{c=1}^{k} \sum_{\mathbf{x}_i \in \pi_c} \mathsf{D}_{\mathbf{w}}(\mathbf{x}_i, \mu_c) + \lambda_1 \sum_{(s,t,\delta) \in \mathcal{P}} \max(\delta - (w_s - w_t), 0) - \lambda_2 \widehat{H}(\mathbf{w}) \tag{3}$$

where $\lambda_1, \lambda_2 \geqslant 0$ are pre-specified parameters. The ratio between λ_1 and λ_2 encodes how confident we're about the feature order preferences. If λ_1 is very small and λ_2 is very large, the clustering model will degenerate to uniform weighting and all the feature order preferences will be ignored.

After simple transformations as done in Eq. (2) using m auxiliary variables $\xi = [\xi_p] \in \mathbb{R}^m$ where $p \in \mathcal{P}$, our overall clustering objective can be written as

$$\begin{aligned} \min_{\{\mathbf{w},\xi\},\{\pi_c\}_{c=1}^k,\{\mu_c\}_{c=1}^k} \quad & \sum_{c=1}^{k} \sum_{\mathbf{x}_i \in \pi_c} \mathsf{D}_{\mathbf{w}}(\mathbf{x}_i, \mu_c) + \lambda_1 \sum_{p \in \mathcal{P}} \xi_p - \lambda_2 \widehat{H}(\mathbf{w}) \\ \text{subjectto} \quad & \mathbf{w} \in \Delta_d \\ & w_s - w_t \geqslant \delta - \xi_p \quad \text{forall} p = (s,t,\delta) \in \mathcal{P} \\ & \xi_p \geqslant 0 \quad \text{forall} p \in \mathcal{P} \end{aligned} \tag{4}$$

2.2. *Parameterized distortion measures*

Various distortion measures can be chosen for the clustering objective. Different distortion measures imply different assumptions about the underlying distribution of the data under consideration. Many useful distortion measures, such as squared Euclidean distance and KL divergence, belong to a broad class of distortion functions known as Bregman divergences [3] which is defined as follows.

Definition 1. *Suppose $\phi : \mathcal{S} \mapsto \mathbb{R}$ is a strictly convex function where $\mathcal{S} \subseteq \mathbb{R}^d$ is a convex set and $\phi(\cdot)$ is differentiable on $\mathrm{ri}(\mathcal{S})$, namely, the relative interior of $\mathcal{S}$. Then Bregman divergence $\mathsf{d}_\phi : \mathcal{S} \times \mathrm{ri}(\mathcal{S}) \mapsto \mathbb{R}_+$ can be defined as follows.*

$$\mathsf{d}_\phi(\mathbf{z}_1, \mathbf{z}_2) = \phi(\mathbf{z}_1) - \phi(\mathbf{z}_2) - \langle \mathbf{z}_1 - \mathbf{z}_2, \nabla\phi(\mathbf{z}_2)\rangle \tag{5}$$

where $\nabla\phi$ is the gradient of ϕ.

Different $\phi(\cdot)$ will lead to diverse divergences. For example, if $\phi(z) = z^2$, then $\mathsf{d}_\phi(z_1, z_2) = (z_1 - z_2)^2$ is the squared loss (square Euclidean distance). If $\phi(z) = z\log(z) - z$, then $\mathsf{d}_\phi(z_1, z_2) = z_1 \log(z_1/z_2) - (z_1 - z_2)$ is generalized I-divergence. Other Bregman divergences include Itakura-Saito distance, KL-divergence, logistic loss, hinge loss and Mahalanobis distance.

An key property of Bregman divergence is formally stated as follows [3].

Lemma 1. *Suppose $\{\mathbf{z}_i\}_{i=1}^{l} \subset \mathcal{S} \subseteq \mathbb{R}^d$ and $\frac{1}{l}\sum_{i=1}^{l} \mathbf{z}_i \in \mathrm{ri}(\mathcal{S})$. For Bregman divergence $\mathsf{d}_\phi : \mathcal{S} \times \mathrm{ri}(\mathcal{S}) \mapsto \mathbb{R}_+$, the following problem*

$$\min_{\mathbf{s}\in\mathrm{ri}(\mathcal{S})} \sum_{i=1}^{l} \mathsf{d}_\phi(\mathbf{z}_i, \mathbf{s}) \tag{6}$$

has a unique solution: $\mathbf{s}^\dagger = \frac{1}{l}\sum_{i=1}^{l} \mathbf{z}_i$.

We use a parameterized version of Bregman divergences for $\mathsf{D}_{\mathbf{w}}(\cdot,\cdot)$ in Eq. (1), specifically,

$$\mathsf{D}_{\mathbf{w}}(\mathbf{x}_i, \mu_c) = \sum_{j=1}^{d} \frac{w_j}{v_j} \mathsf{d}_\phi(x_{ij}, \mu_{cj}) \tag{7}$$

where $\mathbf{v} = [v_1, \ldots, v_d]^\top \in \mathbb{R}^d$, which is used to scale the within-cluster distortion in each dimension to $[0, 1]$, is defined as follows.

$$v_j = \min_u \sum_{i=1}^{n} \mathsf{d}_\phi(x_{ij}, u) \tag{8}$$

According to Lemma 1, $v_j = \sum_{i=1}^{n} \mathsf{d}_\phi(x_{ij}, \bar{\mu}_j)$ where $\bar{\mu} = [\bar{\mu}_1, \ldots, \bar{\mu}_d]^\top = \frac{1}{n}\sum_{i=1}^{n} \mathbf{x}_i$ is the global mean of the data points.

3. Algorithm derivation

In this section, we derive an efficient algorithm to optimize the clustering problem in Eq. (4). Combined with $\widehat{H}(\mathbf{w}) = 1 - \mathbf{w}^\top\mathbf{w}$ and Eq. (7), the clustering objective (4) is equivalent to the following one.

$$\begin{aligned} \min_{\{\mathbf{w},\xi\},\{\pi_c\}_{c=1}^{k},\{\mu_c\}_{c=1}^{k}} \quad & \sum_{c=1}^{k} \sum_{\mathbf{x}_i\in\pi_c} \sum_{j=1}^{d} \frac{w_j}{v_j} \mathsf{d}_\phi(x_{ij}, \mu_{cj}) + \lambda_1 \sum_{p\in\mathcal{P}} \xi_p + \lambda_2 \mathbf{w}^\top\mathbf{w} \\ \text{subjectto} \quad & \mathbf{w} \in \Delta_d \\ & w_s - w_t \geqslant \delta - \xi_p \quad \text{forall} p = (s, t, \delta) \in \mathcal{P} \\ & \xi_p \geqslant 0 \quad \text{forall} p \in \mathcal{P} \end{aligned} \tag{9}$$

This is the clustering objective we want to optimize in this section. Extensions will be discussed in the next section. Note that the regularization term $-\widehat{H}(\mathbf{w})$ is very important to our formulation. For example, in the extreme case that no preferences are available ($m = 0$), if $\lambda_2 = 0$, then the feature with the smallest intra-cluster distortion will receive weight 1 and others get zero weight, which is obviously undesirable in practice.

In Eq. (9), there're 3 sets of unknown variables, namely, $\{\mathbf{w}, \xi\}$, $\{\pi_c\}_{c=1}^k$ and $\{\mu_c\}_{c=1}^k$. When two of them are fixed, the subproblem of computing the optimal values for the variables in the remaining set is easy to solve. Hence, problem (9) can be solved by iteratively updating $\{\mathbf{w}, \xi\}$, $\{\pi_c\}_{c=1}^k$ and $\{\mu_c\}_{c=1}^k$ so that the objective value gradually decreases. This approach can be thought of as a "block coordinate descent" method [6].

3.1. The computation of $\{\pi_c\}_{c=1}^k$ for given $\{\mu_c\}_{c=1}^k$ and $\{\mathbf{w}, \xi\}$

Given an existing set of cluster representatives $\{\mu_c\}_{c=1}^k$ and $\{\mathbf{w}, \xi\}$, computing the optimal clustering $\{\pi_c\}_{c=1}^k$ in problem (9) is equivalent to solving the following minimization problem

$$\min_{\{\pi_c\}_{c=1}^k} \quad \sum_{c=1}^{k} \sum_{\mathbf{x}_i \in \pi_c} \mathsf{D}_{\mathbf{w}}(\mathbf{x}_i, \mu_c) \tag{10}$$

where $\mathsf{D}_{\mathbf{w}}(\mathbf{x}_i, \mu_c) = \sum_{j=1}^d \frac{w_j}{v_j} \mathsf{d}_\phi(x_{ij}, \mu_{cj})$. Therefore, each data point $\mathbf{x}_i$ should be assigned to a cluster π_c so that $\mathsf{D}_{\mathbf{w}}(\mathbf{x}_i, \mu_c)$ is minimized (ties are resolved arbitrarily). After the cluster assignment, we obtain

$$\pi_c = \{\mathbf{x} \in \{\mathbf{x}_i\}_{i=1}^n \,|\, \mathsf{D}_{\mathbf{w}}(\mathbf{x}, \mu_c) \leqslant \mathsf{D}_{\mathbf{w}}(\mathbf{x}, \mu_l) \mathrm{forall} 1 \leqslant l \leqslant k\}\,. \tag{11}$$

3.2. The computation of $\{\mu_c\}_{c=1}^k$ for given $\{\pi_c\}_{c=1}^k$ and $\{\mathbf{w}, \xi\}$

Given an existing clustering $\{\pi_c\}_{c=1}^k$ and $\{\mathbf{w}, \xi\}$, computing the optimal cluster representatives $\{\mu_c\}_{c=1}^k$ in problem (9) is equivalent to solving the following minimization problem

$$\min_{\{\mu_c\}_{c=1}^k} \quad \sum_{c=1}^{k} \sum_{\mathbf{x}_i \in \pi_c} \sum_{j=1}^{d} \frac{w_j}{v_j} \mathsf{d}_\phi(x_{ij}, \mu_{cj}) = \sum_{c=1}^{k} \sum_{j=1}^{d} \frac{w_j}{v_j} g(\mu_{cj}) \tag{12}$$

where $g(\mu_{cj}) = \sum_{\mathbf{x}_i \in \pi_c} \mathsf{d}_\phi(x_{ij}, \mu_{cj})$. Since $w_j, v_j \geqslant 0$, problem (12) is equivalent to minimizing $g(\mu_{cj})$ for each μ_{cj} where $1 \leqslant c \leqslant k$ and $1 \leqslant j \leqslant d$. According to Lemma 1, $\mu_{cj} = \frac{1}{n_c} \sum_{\mathbf{x}_i \in \pi_c} x_{ij}$ is the minimizer of $g(\mu_{cj})$. Therefore, the optimal value of problem (12) is achieved when $\mu_c = \frac{1}{n_c} \sum_{\mathbf{x}_i \in \pi_c} \mathbf{x}_i$.

3.3. The computation of $\{\mathbf{w}, \xi\}$ for given $\{\pi_c\}_{c=1}^k$ and $\{\mu_c\}_{c=1}^k$

Given an existing clustering $\{\pi_c\}_{c=1}^k$ and cluster representatives $\{\mu_c\}_{c=1}^k$, computing the optimal $\{\mathbf{w}, \xi\}$ in problem (9) is equivalent to solving the following minimization problem

$$\begin{aligned} \min_{\mathbf{w} \in \mathbb{R}^d,\, \xi \in \mathbb{R}^m} \quad & \mathbf{w}^\top \mathbf{b} + \xi^\top \mathbf{u} + \frac{1}{2} \mathbf{w}^\top \mathbf{w} \\ \mathrm{subjectto} \quad & \mathbf{w} \in \Delta_d \\ & \mathbf{A}\mathbf{w} + \xi \geqslant \delta \\ & \xi \geqslant 0 \end{aligned} \tag{13}$$

where $\mathbf{u} = \frac{\lambda_1}{2\lambda_2}\mathbf{1}_m \in \mathbb{R}^m$, $\mathbf{b} = [b_1, b_2, \cdots, b_d]^\top \in \mathbb{R}^d$ and

$$b_j = \frac{1}{2\lambda_2}\sum_{c=1}^{k}\sum_{\mathbf{x}_i\in\pi_c}\frac{1}{v_j}\mathsf{d}_\phi(x_{ij}, \mu_{cj})$$

Besides, $\mathbf{A} = [a_{ij}] \in \mathbb{R}^{m\times d}$ and

$$a_{ij} = \begin{cases} 1 & \text{if} \quad s_i = j \\ -1 & \text{if} \quad t_i = j \\ 0 & \text{otherwise} \end{cases}$$

where $p_i = (s_i, t_i, \delta_i)$ is the i-th feature order preference.

Problem (13) is a (convex) quadratic programming problem [7] with $d + m$ variables, $d + 2m$ linear inequality constraints and 1 linear equality constraint.

The Lagrange dual problem [7] associated with problem (13) can be derived and simplified as follows.

$$\begin{aligned} \min_{\alpha\in\mathbb{R}^m,\ \beta\in\mathbb{R}^d} \quad & \frac{1}{2}\cdot(\alpha^\top,\ \beta^\top)\left\{\begin{pmatrix}\mathbf{A}\\ \mathbf{I}_d\end{pmatrix}\begin{pmatrix}\mathbf{A}^\top,\ \mathbf{I}_d\end{pmatrix} - \begin{pmatrix}\mathbf{0} & \mathbf{0}\\ \mathbf{0} & \frac{1}{d}\mathbf{1}_d\mathbf{1}_d^\top\end{pmatrix}\right\}\begin{pmatrix}\alpha\\ \beta\end{pmatrix} \\ & - \left(\mathbf{b}^\top - \frac{1}{d}\mathbf{b}^\top\mathbf{1}_d\mathbf{1}_d^\top - \frac{1}{d}\mathbf{1}_d^\top\right)\begin{pmatrix}\mathbf{A}^\top,\ \mathbf{I}_d\end{pmatrix}\begin{pmatrix}\alpha\\ \beta\end{pmatrix} - \alpha^\top\delta \\ \text{subject to} \quad & 0 \leqslant \alpha \leqslant \mathbf{u} \\ & 0 \leqslant \beta \end{aligned} \tag{14}$$

Problem (14) is a bound-constrained quadratic optimization problem, which can be solved efficiently using the gradient projection method [22]. In each iteration of the gradient projection method, a step size has to be computed along a piecewise-linear path with $O(m + d)$ line segments. For each line segment, the time complexity is dominated by calculating $\mathbf{A}\mathbf{g}_d$ and $\mathbf{g}_m\mathbf{A}$ where $\mathbf{g}_d \in \mathbb{R}^d$ and $\mathbf{g}_m \in \mathbb{R}^m$ are vectors arising in the procedure. Since matrix $\mathbf{A}$ has only $O(m)$ non-zero elements, searching each line segment takes time $O(m + d)$ which makes the time complexity of each iteration $O\left((m + d)^2\right)$.

When the optimal solution of the dual problem (14) is obtained, the optimal solution of the primal problem (13) can be calculated as follows.

$$\mathbf{w} = \mathbf{A}^\top\alpha + \left(\beta - \frac{1}{d}\mathbf{1}_d\mathbf{1}_d^\top\beta\right) - \left(\mathbf{b} - \frac{1}{d}\mathbf{1}_d\mathbf{1}_d^\top\mathbf{b} - \frac{1}{d}\mathbf{1}_d\right)$$

This can be verified using the KKT condition [7].

In [20], the criterion for selecting the optimal feature weighting in clustering makes the feature weights difficult to determine. In fact, they calculate the weights through an exhaustive search over a coarse grid on Δ_d. In practice, their method can only determine the approximately optimal values of a few weights. Instead, our problem formulation makes the subproblem of determining feature weights much easier to solve. Furthermore, the incorporation of domain knowledge such as feature order preferences becomes natural. Even with the generalized entropy discussed in Section 4.2, this subproblem for computing $\mathbf{w}$ is still a convex programming problem in which any locally optimal solution is also globally optimal. There're very effective algorithms that can solve convex programs reliably and efficiently [7].

```
Algorithm: CFP ($\{\mathbf{x}_i\}_{i=1}^n$, $k$, $\mathcal{P}$, $\lambda_1$, $\lambda_2$)
Input: Dataset $\{\mathbf{x}_i\}_{i=1}^n$, number of output clusters $k$, a set of feature order pref-
    erences $\mathcal{P}$, parameters $\lambda_1$ and $\lambda_2$.
Output: Output clustering $\{\pi_c\}_{c=1}^k$.
Procedure:
 1. Initialize $k$ cluster representatives $\{\boldsymbol{\mu}_c\}_{c=1}^k$ and set $\mathbf{w} = [\frac{1}{d}, \cdots, \frac{1}{d}]^T$;
 2. repeat
    2a. E-step : Given $\{\boldsymbol{\mu}_c\}_{c=1}^k$ and $\{\mathbf{w}\}$, re-assign data points to clusters as in
        Section 3.1, thus obtaining $\{\pi_c\}_{c=1}^k$.
    2b. M-step1: Given $\{\pi_c\}_{c=1}^k$, re-calculate cluster representatives $\{\boldsymbol{\mu}_c\}_{c=1}^k$ as
        in Section 3.2;
    2c. M-step2: Given $\{\pi_c\}_{c=1}^k$ and $\{\boldsymbol{\mu}_c\}_{c=1}^k$, re-compute $\{\mathbf{w}, \boldsymbol{\xi}\}$ by solving the
        quadratic programming problem (13);
    until convergence;
 3. return $\{\pi_c\}_{c=1}^k$;
```

Fig. 1. CFP **algorithm**.

3.4. *The main algorithm*

The outline of our algorithm for Clustering with Feature order Preferences (CFP) is presented in Fig. 1. The "convergence" criterion is met when the change in the clustering objective value between two successive iterations is less than some pre-specified threshold. In our experiments, our algorithm typically converges within less than 50 iterations.

As discussed in the previous subsections, the iterative updating procedure of CFP decreases the objective value in problem (9) after each iteration. Besides, the objective value is bounded below by zero. Therefore, the algorithm CFP converges to a locally optimal solution in a finite number of steps.

3.5. *Computational complexity*

In this section, we analyze the computational complexity of the proposed algorithm. It can be easily seen that the time complexity of `E-step` and `M-step1` are $O(nkd)$ and $O(nd)$ respectively. For `M-step2` which is the feature weight learning algorithm, the time complexity is $O\left(L_2(m+d)^2\right)$ where L_2 is the number of iterations in the gradient projection method. If L_1 is the number of iterations of the overall algorithm. The overall complexity is $O\left(L_1\left(nkd + L_2(m+d)^2\right)\right)$.

4. Extensions and discussion

4.1. *Extension to other distortion measures*

Our clustering framework can be extended to other types of distortion (or similarity) measures including directional similarity functions such as cosine similarity and Pearson's correlation [16,5]. We use cosine similarity as an example to explain the extension. We define parameterized cosine distortion as follows.

$$\mathsf{D}_{\mathbf{w}}(\mathbf{x}, \mathbf{y}) = 1 - \frac{\langle \mathbf{x}, \mathbf{y} \rangle_{\mathbf{w}}}{\|\mathbf{x}\|_{\mathbf{w}} \, \|\mathbf{y}\|_{\mathbf{w}}} \tag{15}$$

where $\langle \mathbf{x}, \mathbf{y}\rangle_{\mathbf{w}} = \sum_{j=1}^{d} w_j x_j y_j$ and $\|\mathbf{x}\|_{\mathbf{w}} = \sqrt{\langle \mathbf{x}, \mathbf{x}\rangle_{\mathbf{w}}}$. The updating procedure for μ_c in `M-step1` of Fig. 1 becomes

$$\mu_c = \frac{\sum_{\mathbf{x}_i \in \pi_c} \mathbf{x}_i}{\left\|\sum_{\mathbf{x}_i \in \pi_c} \mathbf{x}_i\right\|_{\mathbf{w}}} \tag{16}$$

Note that if other distortion measures are used, the subproblem of computing the feature weights given $\{\pi_c\}_{c=1}^{k}$ and $\{\mu_c\}_{c=1}^{k}$ may not be a convex programming problem any more, so locally optimal solution to the subproblem of feature weight learning may not be globally optimal.

4.2. *Extension to generalized entropy*

To formally give the definition of generalized entropy, we first briefly introduce the definition of concave function as follows [23].

Definition 2. *Suppose $X \subseteq \mathbb{R}^d$ is a convex set, $f : X \mapsto \mathbb{R}$ is concave if and only if for any $\mathbf{x}_1, \mathbf{x}_2 \in X$ and $\theta \in (0, 1)$,*

$$f(\theta \mathbf{x}_1 + (1-\theta)\mathbf{x}_2) \geqslant \theta f(\mathbf{x}_1) + (1-\theta) f(\mathbf{x}_2).$$

Generalized entropy measures the degree of uncertainty or impurity within a probability distribution. Here we only consider the discrete probability distribution represented by the probability simplex Δ_d. Each vector $\mathbf{w} \in \Delta_d$ corresponds to a probability distribution on a set of d elements, with w_j interpreted as the probability of the j-th element. A recently proposed definition of generalized entropy is formulated as follows [19].

Definition 3. We define *generalized entropy* as a mapping

$$\widehat{H} : \Delta_d \mapsto \mathbb{R}_+$$

that satisfies the following two criteria (symmetry and concavity):

1. For any $\mathbf{w}_1 \in \Delta_d$, and any $\mathbf{w}_2 \in \Delta_d$ whose elements are a permutation of the elements of $\mathbf{w}_1$, $\widehat{H}(\mathbf{w}_1) = \widehat{H}(\mathbf{w}_2)$.
2. $\widehat{H}(\cdot) \mathrm{is a concave function.}$

A key intuition in this definition is that the more uniform the elements of $\mathbf{w} \in \Delta_d$ are, the larger the value of generalized entropy becomes. The motivation and detailed derivation for this definition is explained in [19]. Here we give some specific examples of generalized entropy as follows:

1. $\widehat{H}(\mathbf{w}) = \sum_{i=1}^{d} -w_i \log(w_i)$, which is the celebrated *Shannon entropy*.
2. Suppose $t > 0$. $\widehat{H}(\mathbf{w}) = t(1 - \sum_{i=1}^{n} w_i^{\beta})$ when $\beta > 1$ and $\widehat{H}(\mathbf{w}) = t(\sum_{i=1}^{n} w_i^{\beta} - 1)$ when 0 $< \beta <$ 1 are generalized entropies [25].
3. $\widehat{H}(\mathbf{w}) = 1 - \mathbf{w}^{\top}\mathbf{w}$, which will be referred to as ℓ_2-*entropy*. This entropy is in essence a special case of the previous definition when $\beta = 2$.
4. $\widehat{H}(\mathbf{w}) = 2 - \sum_{i=1}^{d} |w_i - \frac{1}{d}|$, which will be referred to as ℓ_1-*entropy*.
5. $\widehat{H}(\mathbf{w}) = 1 - \max_{1 \leqslant i \leqslant d} w_i$, which will be referred to as ℓ_∞-*entropy*.

Many other entropies which are special cases of this definition have been proposed in the literature [19]. Since this definition of entropy is very general, our framework can lead to many instantiations. In our proposed algorithm in Section 3, we use ℓ_2-entropy and so the optimization problem in `M-step2` of Fig. 1 is a quadratic programming problem. When ℓ_1-entropy is used, the optimization problem in `M-step2` can be formulated as a linear programming problem. With distortion measure (7), whichever entropy is used, the optimization problem in `M-step2` will always be a convex programming problem, since $\widehat{H}(\cdot)$ is a concave function and all the constraints in the clustering problem (9) are linear.

4.3. Extension to multiple, heterogeneous feature spaces

Our framework can be directly extended to multiple, heterogeneous feature spaces [20]. Each object is represented by a tuple of d component feature vectors, specifically, $\mathbf{x} = (\mathbf{x}^{(1)}, \cdots, \mathbf{x}^{(d)})$ where $\mathbf{x}^{(j)}$ comes from the j-th feature space, which is associated with a weight w_j. Here, the order preference (s, t, δ) with $\delta > 0$ means that "feature space s is more important than feature space t". The distortion measure in Eq. (7) becomes

$$\mathsf{D}_{\mathbf{w}}(\mathbf{x}_i, \mu_c) = \sum_{j=1}^{d} \frac{w_j}{v_j} \mathsf{d}_\phi(\mathbf{x}_i^{(j)}, \mu_c^{(j)}) \tag{17}$$

The corresponding clustering algorithm can be easily derived.

4.4. Extensions to discrete data

In the clustering model proposed above, we assume that the data points are sampled from $\mathbb{R}^d$, that is, the attributes of the data points are numeric. Here, we'll explain how the model can be modified to deal with discrete data with nominal attributes, namely, categorical data where the category labels are unordered.

Instead of minimizing distortion w.r.t. the cluster representative, we want the data in each cluster and dimension (for each cluster-feature pair) to be as pure as possible. By using an entropy criterion for each cluster-feature pair, we have the overall clustering goal, which is to minimize the following clustering objective function:

$$\frac{1}{k} \sum_{c=1}^{k} \frac{|\pi_c|}{n} \sum_{j=1}^{d} \frac{w_j}{v_j} \widehat{H}(\mathbf{y}_{cj}) + \lambda_1 \sum_{(s,t,\delta)\in\mathcal{P}} \max(\delta - (w_s - w_t),\, 0) - \lambda_2\, \widehat{H}(\mathbf{w}) \tag{18}$$

where $\mathbf{y}_{cj}$ is the distribution of category labels for each cluster c and feature j. Specifically,

$$\mathbf{y}_{cj} = \left(\frac{g_{cj1}}{|\pi_c|}, \frac{g_{cj2}}{|\pi_c|}, \cdots, \frac{g_{cjh_j}}{|\pi_c|} \right)^\top \tag{19}$$

where h_j is the number of category labels in dimension j, and g_{cjl} $(1 \leqslant l \leqslant h_j)$ is the number of the l-th category label in the data w.r.t. the cluster-feature pair cj (see the example in the next paragraph). v_j is a value used to normalize the entropy value in dimension j. Note that different entropies can be used for the first and second $\widehat{H}(\cdot)$ in Eq. (18).

For example, in a 1-dimensional data set $\{a, b, b, b, a\}$ with 5 data points, there're two category labels, namely a and b. Thus, each $\mathbf{y}_{cj}$ is 2-dimensional. If the data set is clustered into 2 clusters $\{a, b, b\}$

Table 1
Summary of datasets

Dataset	*iris*	*optdigits*	*pgblocks*	*pendigits*	*vowel*	*wdbc*
n	150	5620	5473	10992	990	569
d	4	64	10	16	10	30
k	3	10	5	10	11	2

and $\{b, a\}$, we have $\mathbf{y}_{1,1} = (\frac{1}{3}, \frac{2}{3})^\top$ and $\mathbf{y}_{2,1} = (\frac{1}{2}, \frac{1}{2})^\top$. If clustered into another 2 clusters $\{a, a\}$ and $\{b, b, b\}$, we have $\mathbf{y}_{1,1} = (1, 0)^\top$ and $\mathbf{y}_{2,1} = (0, 1)^\top$. Clearly, the latter clustering has a lower value of entropy, thus a better clustering.

A similar iterative algorithm can be derived for optimizing the clustering objective (18) as in Section 3. If different generalized entropies are adopted for the term $\widehat{H}(\mathbf{y}_{cj})$ in Eq. (18), we can obtain different categorical data clustering criteria. Here we briefly explain some of the variants and their relations with previous works on categorical clustering in the literature.

If the ℓ_∞-entropy defined in Section 4.2 is used for the term $\widehat{H}(\mathbf{y}_{cj})$ in Eq. (18), $\lambda_1 = \lambda_2 = 0$, $\mathbf{v} = \mathbf{1}^d$ and the value of $\mathbf{w}$ is fixed and uniform, then objective function (18) is what the k-Modes [12] algorithm aims to optimize. Besides, it is also the criterion function implicitly optimized by the Iterative Voting Consensus (IVC) algorithm in [21].

If the ℓ_2-entropy defined in Section 4.2 is used for the term $\widehat{H}(\mathbf{y}_{cj})$ in Eq. (18), $\lambda_1 = \lambda_2 = 0$, $\mathbf{v} = \mathbf{1}^d$ and the value of $\mathbf{w}$ is fixed and uniform, then objective function (18) is essentially the definition of *category utility* [11].

If the Shannon entropy is used for the term $\widehat{H}(\mathbf{y}_{cj})$ in Eq. (18), $\lambda_1 = \lambda_2 = 0$, $\mathbf{v} = \mathbf{1}^d$ and the value of $\mathbf{w}$ is fixed and uniform, then objective function (18) is the clustering criterion that's been used in [1,4].

5. Experiments

In this section, we present an empirical evaluation of our clustering method with different number of feature order preferences, parameter values and distortion measures on a number of datasets. First, we briefly introduce the basic information of the datasets. We use six datasets from the UCI machine learning repository [2]. Table 1 summarizes the basic properties of the datasets. These datasets provide a good representation of different data distributional characteristics. Note that in all the experiments, the "true" number of clusters k is provided to the clustering algorithms.

5.1. Experimental setting

In practice, feature order preferences would be provided by domain experts. However, for convenience, we generate simulated feature order preferences by using the ground truth class information in our experiments. We first calculate the *within-class distortion* for each dimension $1 \leqslant j \leqslant d$, which is defined as

$$\Theta_j = \frac{1}{v_j} \sum_{c=1}^{k} \sum_{\mathbf{x}_i \text{ inclassc}} \mathsf{d}_\phi(x_{ij}, \mu_{cj}) \tag{20}$$

where μ_c is the centroid of class c. Then, for each dimension $1 \leqslant j \leqslant d$, we calculate the inverse within-class distortion $\Gamma_j = \frac{\sum_{l \neq j} \Theta_l}{\Theta_j}$. After that, we estimate the optimal feature weights by $\widetilde{w}_j = \frac{\Gamma_j}{\sum_{l=1}^{d} \Gamma_l}$.

The weight vector $\widetilde{\mathbf{w}}$ is generally not the optimal weighting for clustering the data at hand. Instead, it is just a rough estimate of the optimal feature weighting. We randomly sample without replacement m pairs (s, t) of features with the constraint that $\widetilde{w}_s$ is among the $\lfloor \frac{d}{2} \rfloor$ largest weights and $\widetilde{w}_t$ is among the $\lfloor \frac{d}{2} \rfloor$ smallest weights. Then our simulated feature order preference is $(s, t, \widetilde{w}_s - \widetilde{w}_t)$.

In our experiments, we use $\widehat{H}(\mathbf{w}) = 1 - \mathbf{w}^\top \mathbf{w}$. Besides, by default, we set the parameters $\lambda_1 = \frac{d}{m}$ and $\lambda_2 = d$ unless otherwise stated. The reason for this is that we want the three terms in Eq. (9) to contribute equally to the objective value. Since the first term is scaled to [0,1] due to $\mathbf{v}$, we want the other terms to be around 1. As the average weight of each feature is $\frac{1}{d}$, the second term is approximately less than $\frac{m}{d}$ and so λ_1 is set to $\frac{d}{m}$. The minimum value of $\mathbf{w}^T\mathbf{w}$ is $\frac{1}{d}$ corresponding to the uniform weighting. We want $\mathbf{w}$ to be as uniform as possible, so λ_2 is set to d.

We compare the performance of our algorithm with Bregman hard clustering [3] which is referred to as BClus. For the cluster initialization, we randomly select k points as the cluster representatives and uniform weighting as the initial weighting $\mathbf{w}$. We set $\phi(x) = x^2$ for BClus and CFP by default. For each run of the considered algorithms (BClus, CFP with different number of feature order preferences), the same 5 sets of initial seeds are used and the clustering result with the best value of objective function is selected for each algorithm. Unless otherwise stated, each performance result is the average of 100 runs of the considered algorithms. The algorithms are implemented in C++.

5.2. *Evaluation criteria*

Given class labels, we adopt two external validity measures, Normalized Mutual Information (*NMI*) [26] and Clustering Accuracy (*Acc*) [29], as our criteria.

Given a clustering $\mathcal{C}$ and the "true" partitioning $\mathcal{B}$ (class labels). The number of clusters in $\mathcal{C}$ and classes in $\mathcal{B}$ are both k. Suppose n_i is the number of objects in the i-th cluster, $n_j^{'}$ is the number of objects in the j-th class and n_{ij} is the number of objects which are in both the i-th cluster and j-th class. *NMI* between $\mathcal{C}$ and $\mathcal{B}$ is calculated as follows [26]:

$$\mathrm{NMI}\,(\mathcal{C}, \mathcal{B}) = \frac{\sum_{i=1}^{k} \sum_{j=1}^{k} n_{ij} \log \frac{n \cdot n_{ij}}{n_i \cdot n_j^{'}}}{\sqrt{\sum_{i=1}^{k} n_i \log \frac{n_i}{n} \sum_{j=1}^{k} n_j^{'} \log \frac{n_j^{'}}{n}}}. \tag{21}$$

Clustering Accuracy (*Acc*) builds a one-to-one correspondence between the clusters and the classes. Suppose the permutation function $\mathrm{Map}(\cdot) : \{i\}_{i=1}^{k} \mapsto \{j\}_{j=1}^{k}$ maps each cluster index to a class index, i.e., Map(i) is the class index that corresponds to the i-th cluster. *Acc* between $\mathcal{C}$ and $\mathcal{B}$ is calculated as follows:

$$\mathrm{Acc}\,(\mathcal{C}, \mathcal{B}) = \frac{\max\left(\sum_{i=1}^{k} n_{i,\mathrm{Map}(i)}\right)}{n} \tag{22}$$

Larger values of *NMI* and *Acc* indicate better clustering performance.

5.3. *Clustering performance*

In this section, we compare the performance of the algorithms. We test our proposed clustering algorithm with various numbers of feature order preferences.

Table 2
Clustering results on the *iris* dataset. "*C.M.*" denotes confusion matrices. CFP(m) is our proposed algorithm with m feature order preferences. This is the clustering result of only a single run of the algorithms. Here $\phi(x) = x^2$. It can be observed that feature order preferences do improve clustering result

Algo.	BClus			CFP(1)			CFP(2)			CFP(3)			CFP(4)		
	50	0	0	**50**	0	0	**50**	0	0	**50**	0	0	**50**	0	0
C.M.	0	**39**	14	0	**47**	13	0	**46**	4	0	**48**	4	0	**48**	4
	0	11	**36**	0	3	**37**	0	4	**46**	0	2	**46**	0	2	**46**
NMI	0.6595			$\nearrow$0.7496			$\nearrow$0.8308			$\nearrow$0.8642			$\rightarrow$0.8642		
Acc	0.8333			$\nearrow$0.8933			$\nearrow$0.9467			$\nearrow$0.9600			$\rightarrow$0.9600		

Table 3
Experimental results on all the datasets. Both *NMI* and *Acc* results are provided here. CFP(m) is our proposed algorithm with m feature order preferences. The results produced by CFP when $m = \lfloor \frac{d}{4} \rfloor, \lfloor \frac{d}{2} \rfloor, d$ are shown

	NMI				*Acc*			
	BClus	CFP$_{d/4}$	CFP$_{d/2}$	CFP$_d$	BClus	CFP$_{d/4}$	CFP$_{d/2}$	CFP$_d$
iris	0.6511	0.7381	0.8265	0.8642	0.8238	0.8913	0.9371	0.9600
optdigits	0.6414	0.6747	0.6897	0.7046	0.6422	0.6831	0.7014	0.7204
pendigits	0.6914	0.6974	0.6968	0.7024	0.6968	0.7079	0.7080	0.7188
pgblocks	0.1194	0.1379	0.1594	0.1820	0.4719	0.5403	0.6015	0.6691
vowel	0.3765	0.3972	0.4109	0.4241	0.3213	0.3362	0.3483	0.3588
wdbc	0.5410	0.6113	0.6182	0.6276	0.9071	0.9225	0.9237	0.9255

First, we consider a small dataset *iris*. Table 2 shows the confusion matrices, *NMI* and *Acc* values obtained by BClus and CFP on the *iris* dataset. The arrows at the left side of *NMI* and *Acc* values indicate whether the value increases ($\nearrow$) or remains unchanged ($\rightarrow$), compared with the algorithm in the previous column. Note that this is the clustering result of only a single run of the algorithms. It can be observed from Table 2 that feature order preferences do improve clustering result for the *iris* dataset. Besides, CFP produces better clustering results as the number of feature order preferences increases. With only a couple of feature order preferences, our algorithm almost recovers the original classes in the *iris* dataset.

The clustering results on all the datasets are shown in Table 3. As for CFP, results with $m = \lfloor \frac{d}{4} \rfloor, \lfloor \frac{d}{2} \rfloor, d$ are shown. As can be seen from Table 3, CFP generally produces better clustering results than BClus, and more feature order preferences often lead to better performance. For each dataset, the best result is often achieved by CFP(d). The results demonstrate that CFP effectively improves clustering quality when some feature order preferences are available.

Then we compare the clustering results of the algorithms on datasets *optdigits* and *wdbc* with many different values of m. The results in terms of *NMI* are shown in Fig. 2. It can be observed that our algorithm CFP consistently outperforms BClus, even with a small number of feature order preferences. With an increasing m, CFP generally produces increasing *NMI* values. Therefore, larger gains in clustering performance can be obtained with more feature order preferences.

5.4. *Different* λ_1 *and* λ_2

We also examine the performance of the considered algorithms when different values of the parameters λ_1 and λ_2 are used. For example, the clustering results for $\lambda_1 = \frac{d}{m}$ and $\lambda_2 = \frac{d}{2}$ are summarized in Table 4 and the clustering results for $\lambda_1 = \frac{d}{2m}$ and $\lambda_2 = \frac{d}{2}$ are summarized in Table 5. From the clustering

Table 4
Experimental results on the datasets. Here, $\lambda_1 = \frac{d}{m}$ and $\lambda_2 = \frac{d}{2}$

	NMI				*Acc*			
	BClus	CFP$_{d/4}$	CFP$_{d/2}$	CFP$_d$	BClus	CFP$_{d/4}$	CFP$_{d/2}$	CFP$_d$
iris	0.6521	0.7581	0.8233	0.8642	0.8251	0.9031	0.9341	0.9600
optdigits	0.6401	0.6749	0.6867	0.7000	0.6400	0.6815	0.6960	0.7143
pendigits	0.6915	0.7012	0.7004	0.7054	0.6974	0.7051	0.7059	0.7178
pgblocks	0.1180	0.1651	0.1720	0.1861	0.4708	0.5908	0.6331	0.6807
vowel	0.3794	0.3964	0.4106	0.4226	0.3263	0.3334	0.3471	0.3571
wdbc	0.5399	0.6108	0.6174	0.6278	0.9069	0.9218	0.9233	0.9254

Table 5
Experimental results on the datasets. Here, $\lambda_1 = \frac{d}{2m}$ and $\lambda_2 = \frac{d}{2}$

	NMI				*Acc*			
	BClus	CFP$_{d/4}$	CFP$_{d/2}$	CFP$_d$	BClus	CFP$_{d/4}$	CFP$_{d/2}$	CFP$_d$
iris	0.6510	0.7587	0.8299	0.8642	0.8235	0.9033	0.9392	0.9600
optdigits	0.6497	0.6782	0.6936	0.7024	0.6535	0.6862	0.7071	0.7183
pendigits	0.6940	0.7009	0.7026	0.7067	0.7019	0.7082	0.7147	0.7262
pgblocks	0.1202	0.1648	0.1799	0.1896	0.4720	0.5943	0.6479	0.6852
vowel	0.3769	0.3968	0.4087	0.4222	0.3234	0.3327	0.3448	0.3558
wdbc	0.5396	0.6106	0.6168	0.6282	0.9068	0.9218	0.9233	0.9256

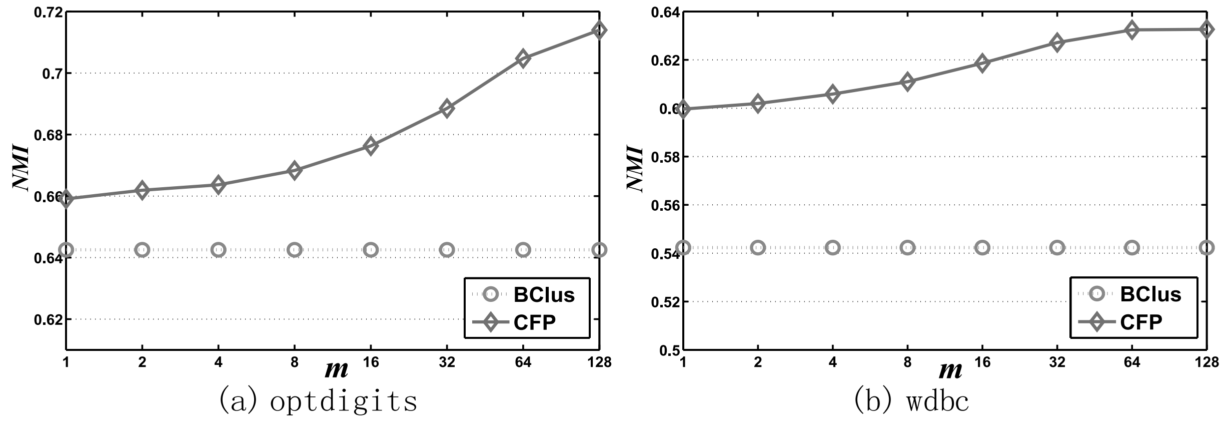

Fig. 2. The clustering performance (*NMI*) on datasets *optdigits* and *wdbc*.

results shown in the two tables, we can see that our algorithm is not very sensitive to the parameters λ_1 and λ_2. As long as they are in a reasonable range, good results can be achieved.

5.5. *Without expert-provided δ in preferences*

In a typical real-world clustering task, an expert only has some vague idea about which features are more important. Therefore, specifying the value of δ_i accurately in a feature order preference (s_i, t_i, δ_i) seems impractical. A more realistic setting is that there're no values of δ_i in the expert-provided feature order preferences. Then the set of feature order preferences becomes $\mathcal{P} = \{(s_i, t_i)\}_{i=1}^m$ instead of $\{(s_i, t_i, \delta_i)\}_{i=1}^m$. In this case, we assume that all the feature order preferences are equally important. By introducing an extra parameter δ, the set of feature order preferences can be transformed into

Table 6
Experimental results on the datasets. Here, $\delta = \frac{1}{d}$ for all the feature order preferences

	NMI				*Acc*			
	BClus	CFP$_{d/4}$	CFP$_{d/2}$	CFP$_{d}$	BClus	CFP$_{d/4}$	CFP$_{d/2}$	CFP$_{d}$
iris	0.6525	0.7085	0.7421	0.7781	0.8251	0.8709	0.8863	0.9008
optdigits	0.6442	0.6858	0.7016	0.7150	0.6478	0.7015	0.7213	0.7441
pendigits	0.6924	0.6919	0.6880	0.6860	0.6968	0.7108	0.7051	0.7007
pgblocks	0.1177	0.1639	0.1905	0.2039	0.4693	0.6085	0.6800	0.7328
vowel	0.3751	0.3996	0.4138	0.4200	0.3210	0.3390	0.3511	0.3572
wdbc	0.5394	0.6059	0.6079	0.6065	0.9067	0.9206	0.9200	0.9183

Table 7
Clustering results on the *iris* dataset. Here, $\phi(x) = x\log(x) - x$. Note that this is the clustering result of only a single run of the algorithms

Algo.	BClus			CFP(1)			CFP(2)			CFP(3)			CFP(4)		
	50	0	0	**50**	0	0	**50**	0	0	**50**	0	0	**50**	0	0
C.M.	0	**34**	12	0	**37**	9	0	**43**	8	0	**46**	4	0	**47**	2
	0	16	**38**	0	13	**41**	0	7	**42**	0	4	**46**	0	3	**48**
NMI	0.6413			$\nearrow$0.6818			$\nearrow$0.7437			$\nearrow$0.8308			$\nearrow$0.8801		
Acc	0.8133			$\nearrow$0.8533			$\nearrow$0.9000			$\nearrow$0.9467			$\nearrow$0.9667		

$\mathcal{P} = \{(s_i, t_i, \delta)\}_{i=1}^m$. Then our proposed clustering algorithm can be applied to the data. Here, only one extra parameter δ has to be pre-specified.

We conduct some experiments on the datasets with feature order preferences without any expert-provided δ_i values. We set the parameter δ to $\delta = \frac{1}{d}$ for all the feature order preferences. The results of the experiments are shown in Table 6. We can see from this Table 6 that even without expert-provided values of δ_i, our algorithm can still benefit from feature order preferences for most datasets. An interesting finding is that on the *pgblocks* dataset, the performance gains achieved by CFP without δ_i values are larger than those achieved by CFP with δ_i values. A possible explanation for this is that the feature weights generated automatically are not optimal. It can also be observed that different clustering performance measures can lead to different results. For example, on the *pendigits* dataset, CFP improves over BClus in terms of *Acc* but has worse *NMI* values.

5.6. Clustering results when $\phi(x) = x\log(x) - x$

In this section, we investigate the clustering performance of the considered algorithms when $\phi(x) = x\log(x) - x$. In this case, generalized I-divergence [3] is actually used in BClus and CFP. Specifically,

$$\mathsf{D}_{\mathbf{w}}(\mathbf{x}_i, \mu_c) = \sum_{j=1}^{d} \frac{w_j}{v_j}\left(x_{ij}\log\frac{x_{ij}}{\mu_{cj}} - (x_{ij} - \mu_{cj})\right) \tag{23}$$

First, we consider the dataset *iris*. Table 7 shows the confusion matrices, clustering performance in terms of *NMI* and *Acc* obtained by BClus and CFP on the *iris* dataset. The notations are the same as Table 2. Again, this is the clustering result of only a single run of the algorithms. As Table 7 shows, feature order preferences will improve the clustering performance for the *iris* dataset. Additionally, CFP produces better clustering results with more feature order preferences provided.

The clustering results with $m = \lfloor\frac{d}{4}\rfloor, \lfloor\frac{d}{2}\rfloor, d$ are displayed in Table 8. Note that $\phi(x) = x\log(x) - x$ can only applied to non-negative data points, thus dataset $vowel$ is omitted from the comparison since

Table 8
Experimental results on the datasets. Here $\phi(x) = x\log(x) - x$

	NMI				*Acc*			
	BClus	CFP$_{d/4}$	CFP$_{d/2}$	CFP$_d$	BClus	CFP$_{d/4}$	CFP$_{d/2}$	CFP$_d$
iris	0.6421	0.6921	0.7729	0.8751	0.8143	0.8607	0.9092	0.9647
optdigits	0.6284	0.6363	0.6469	0.6585	0.6236	0.6364	0.6473	0.6600
pendigits	0.6802	0.6875	0.6916	0.6953	0.6705	0.6734	0.6774	0.6828
pgblocks	0.1695	0.1713	0.1741	0.1778	0.4510	0.4965	0.4791	0.4590
wdbc	0.5666	0.5970	0.6127	0.6368	0.9156	0.9233	0.9270	0.9323

it has negative data values. Similar conclusions can be drawn from the comparison results in Table 8. Generally, CFP produces better clustering results than BClus. With more feature order preferences, better performance can be achieved. The clustering results in Table 8 have confirmed that performance boost can be gained by effectively utilizing some feature order preferences. An exception is that in the *pgblocks* dataset, the *NMI* values increase but the *Acc* values fluctuates. Thus, we adopt another clustering performance measure called *AE* (*Average Entropy*) [30]. Lower values of *AE* indicate better clusterings. The resultant *AE* values are 0.1980, 0.1981, 0.1964, 0.1943. Considering the three performance measures together, we can still see that more feature order preferences generally lead to better performance.

6. Conclusions and future work

In this paper, we propose a clustering model that takes into account feature order preferences effectively. Our clustering objective integrates feature weight learning into prototype-based clustering. We discuss how to extend our model to deal with different distortion measures and discrete data. Experimental results show that our proposed algorithm effectively improves clustering quality.

Many directions can be pursued for potential future work, some of which are listed as follows.

- First, it would be interesting to investigate automatic parameter selection method for λ_1 and λ_2 instead of using pre-specified values. It is reasonable that if more more feature order preferences are present or the feature order preferences are very reliable, then λ_2 can be set to have a relatively lower value.
- Second, listwise feature order preferences can be used instead of pairwise ones. If directly transformed into pairwise constraints, there'll be a quadratic number of pairwise constraints which will hurt the computational efficiency. Therefore, more efficient solutions have to be derived.
- Third, it might be fruitful to incorporate feature order preferences into the process of cluster initialization. Prototype-based clustering algorithms such as $k$means are very sensitive to initial cluster centroids. Therefore, utilizing feature order preferences can hopefully mitigate the problem of poor initialization.
- Furthermore, it's also interesting to investigate casting our model into a probabilistic framework, and combine clustering with instance-level constraints and feature order preferences in this unifying framework.

Acknowledgements

This work is supported in part by NSFC grants 60673103 and 60721061.

References

[1] P. Andritsos, P. Tsaparas, R.J. Miller and K.C. Sevcik, LIMBO: Scalable clustering of categorical data. In *Proceedings of the 9th International Conference on Extending Database Technology* (*EDBT*), 2004.

[2] A. Asuncion and D. Newman, *UCI Machine Learning Repository,* University of California, Irvine, School of Information and Computer Sciences, 2007.

[3] A. Banerjee, S. Merugu, I.S. Dhillon and J. Ghosh, Clustering with bregman divergences, *Journal of Machine Learning Research* **6** (2005), 1705–1749.

[4] D. Barbará, Y. Li and J. Couto, COOLCAT: an entropy-based algorithm for categorical clustering. In *Proceedings of the 11th ACM International Conference on Information and Knowledge Management* (*CIKM*), 2002.

[5] S. Basu, M. Bilenko and R.J. Mooney, A probabilistic framework for semi-supervised clustering. In *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (*KDD*), 2004, pages 59–68.

[6] D.P. Bertsekas, *Nonlinear Programming,* Athena Scientific, 2nd edition, 1999.

[7] S. Boyd and L. Vandenberghe, *Convex Optimization*, Cambridge University Press, 2004.

[8] C.J.C. Burges, T. Shaked, E. Renshaw, A. Lazier, M. Deeds, N. Hamilton and G.N. Hullender, Learning to rank using gradient descent. In *Proceedings of the Twenty-Second International Conference on Machine Learning* (*ICML*), 2005, pages 89–96.

[9] C. Domeniconi, D. Gunopulos, S. Ma, B. Yan, M. Al-Razgan and D. Papadopoulos, Locally adaptive metrics for clustering high dimensional data, *Data Mining and Knowledge Discovery* **14**(1) (2007), 63–97.

[10] V. Estivill-Castro, Why so many clustering algorithms – a position paper, *SIGKDD Explorations* **4**(1) (2002), 65–75.

[11] D. H. Fisher, Knowledge acquisition via incremental conceptual clustering, *Machine Learning* **2**(2) (1987), 139–172.

[12] Z. Huang, Extensions to the k-means algorithm for clustering large data sets with categorical values, *Data Mining and Knowledge Discovery* **2**(3) (1998), 283–304.

[13] A.K. Jain and R.C. Dubes, *Algorithms for Clustering Data,* Prentice Hall, 1988.

[14] A.K. Jain, M.N. Murty and P.J. Flynn, Data clustering: A review, *ACM Computing Surveys* **31**(3) (1999), 264–323.

[15] P. Jain, R. Meka and I.S. Dhillon, Simultaneous unsupervised learning of disparate clusterings. In *Proceedings of the SIAM International Conference on Data Mining* (*SDM*), 2008, pages 858–869.

[16] P.E.J. Kanti and V. Mardia, *Directional Statistics,* John Wiley and Sons Ltd., 2nd edition, 2000.

[17] B. Kulis, S. Basu, I.S. Dhillon and R.J. Mooney, Semi-supervised graph clustering: a kernel approach. In *Proceedings of the Twenty-Second International Conference on Machine Learning* (*ICML*), 2005, pages 457–464.

[18] Y. Liu, R. Jin and A.K. Jain, Boostcluster: boosting clustering by pairwise constraints. In *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (*KDD*), 2007, pages 450–459.

[19] P. Luo, G. Zhan, Q. He, Z. Shi and K. Lü, On defining partition entropy by inequalities, *IEEE Transactions on Information Theory* **53**(9) (2007), 3233–3239.

[20] D.S. Modha and W.S. Spangler, Feature weighting in *k*-means clustering, *Machine Learning* **52**(3) (2003), 217–237.

[21] N. Nguyen and R. Caruana, Consensus clustering, In *Proceedings of the 7th IEEE International Conference on Data Mining* (*ICDM*), 2007.

[22] J. Nocedal and S.J. Wright, *Numerical Optimization,* Springer Verlag, 2nd edition, 2006.

[23] R.T. Rockafellar, *Convex Analysis,* Princeton University Press, 1970.

[24] J. Shi and J. Malik, Normalized cuts and image segmentation, *IEEE Transactions on Pattern Analysis and Machine Intelligence* **22**(8) (2000), 888–905.

[25] D.A. Simovici and S. Jaroszewicz, An axiomatization of partition entropy, *IEEE Transactions on Information Theory* **48**(7) (2002), 2138–2142.

[26] A. Strehl and J. Ghosh, Cluster ensembles – a knowledge reuse framework for combining multiple partitions, *Journal of Machine Learning Research* **3** (2002), 583–617.

[27] W. Tang, H. Xiong, S. Zhong and J. Wu, Enhancing semi-supervised clustering: a feature projection perspective. In *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (*KDD*), 2007, pages 707–716.

[28] F. Wang, C. Zhang and T. Li, Regularized clustering for documents. In *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (*SIGIR*), 2007, pages 95–102.

[29] M. Wu and B. Schölkopf, A local learning approach for clustering. In *Advances in Neural Information Processing Systems 19* (*NIPS*), 2006, pages 1529–1536.

[30] Y. Zhao and G. Karypis, Empirical and theoretical comparisons of selected criterion functions for document clustering, *Machine Learning* **55**(3) (2004), 311–331.

[31] X. Zhu and A. Goldberg, Kernel regression with order preferences. In *Proceedings of the Twenty-Second AAAI Conference on Artificial Intelligence* (*AAAI*), 2007.