Principal Direction Divisive Partitioning with kernels and *k*-means steering

Dimitrios Zeimpekis[1] and Efstratios Gallopoulos[2]

[1] Department of Computer Engineering and Informatics, University of Patras, Greece dsz@hpclab.ceid.upatras.gr
[2] Department of Computer Engineering and Informatics, University of Patras, Greece stratis@hpclab.ceid.upatras.gr

Summary. Clustering is a fundamental task in data mining. We propose, implement and evaluate several schemes that combine partitioning and hierarchical algorithms, specifically k-means and Principal Direction Divisive Partitioning (PDDP). Using available theory regarding the solution of the clustering indicator vector problem, we use 2-means to induce partitionings around fixed or varying cut-points. 2-means is applied either on the data or over its projection on a one-dimensional subspace. These techniques are also extended to the case of PDDP(l), a multiway clustering algorithm generalizing PDDP. To handle data that does not lend itself to linear separability, the algebraic framework is established for a kernel variant, KPDDP. Extensive experiments demonstrate the performance of the above methods and suggest that it is advantageous to steer PDDP using k-means. It is also shown that KPDDP can provide results of superior quality than kernel k-means.

Keywords: Divisive clustering, PDDP, spectral clustering, k-means, kernel methods.

1 Introduction

Clustering is a major operation in Text Mining (TM) and in myriad other applications. We consider algorithms that assume the vector space representation for data objects [25], modeled as feature-object matrices (term-document matrices in TM, hereafter abbreviated as tdm's) so that data collections are represented as $m \times n$ matrices A, where a_{ij} measures the importance of term i in document j. Two broad categories of clustering algorithms are partitional (the best known being k-means) and hierarchical, the latter being very desirable for Web type applications[4].

k-means models clusters by means of their centroids. Starting from some initial guess for the k centroids, say $\{c_i\}_{i=1}^{k}$, representing the clusters, it iterates by first reclustering data depending on their distance from the current set

of centroids, and then updating the centroids based on the new assignments. The progress of the algorithm can be evaluated using the objective function $\sum_{i=1}^{k} \sum_{a_j \in \text{cluster}(i)} d(c_i, a_j)$, where d is some distance metric, e.g. quadratic Euclidean distance [18]. This minimization problem for the general case is NP-hard. Two well known disadvantages of the algorithm are that the generated clusters depend on the specific selection of initial centroids (e.g. random, in the absence of any data or application related information) and that the algorithm can be trapped at local minima of the objective function. Therefore, one run of k-means can easily lead to clusters that are not satisfactory and users are forced to initialize and run the algorithm multiple times.

On the other hand, a highly desirable feature especially for Web-based applications, is the clustering structure imposed by hierarchical clustering algorithms (divisive or agglomerative), such as Single Link [30]. Bisecting k-means is a divisive hierarchical variant of k-means that constructs binary hierarchies of clusters. The algorithm starts from a single cluster containing all data points, and at each step selects a cluster and partitions it into two subclusters using the classical k-means for the special case of $k = 2$. Unfortunately, the weaknesses of k-means (cluster quality dependent upon initialization and trappings at local minima) remain.

One particularly powerful class of clustering algorithms, with origins in graph partitioning ([14]), utilizes spectral information from the underlying tdm[3]. These algorithms provide the opportunity for the deployment of computational technologies from numerical linear algebra, an area that has seen enormous expansion in recent decades. In this paper we focus on Principal Direction Divisive Partitioning (PDDP), a clustering algorithm from this class that can be interpreted using Principal Component Analysis (PCA). The algorithm was proposed by D. Boley; ref. [7] is the primary source. PDDP can be quite effective when clustering text collections as it exploits sparse matrix technology and iterative algorithms for very large eigenvalue problems. See [8, 22, 27, 33] as well as the recent monograph [18] and references therein that present several variants of PDDP. At each iteration, PDDP selects one of the current clusters and partitions it into 2 subclusters using information from the PCA of the corresponding tdm. In fact, the basic PDDP algorithm can be viewed as a special case of a more general algorithm, PDDP(l), that constructs 2^l-ary cluster trees. PDDP is known to be an effective clustering method for TM, where tdm's are very large and extremely sparse. The algorithm requires the repeated extraction of leading singular vector information from the tdm[4] and submatrices thereof via the singular value decomposition (SVD). Even though, in general, the SVD is an expensive computation, it has long been known that significant savings can result when seeking only selected singular

[3] See Chris Ding's collection of tutorials and references in `https://crd.lbl.gov/~cding/Spectral`.

[4] More precisely, from the matrix resulting from the tdm after centering it by subtracting from each column their centroid.

triplets from very sparse tdm's [5]. PDDP, however, is typically more expensive than k-means. Furthermore, the partitioning performed at each step of PDDP is duly determined by the outcome of the partial SVD of the cluster tdm (cf. Section 2). Despite the convenient deterministic nature of this step, it is easy to construct examples where PDDP produces inferior partitionings than k-means.

For some time now, we have been studying the characteristics of PDDP and have been considering ways to improve its performance. An early contribution in this direction was PDDP(l) ([33]), a generalization along the lines of early work in multiway spectral graph partitioning [1, 2, 16]. Our first original contribution here is to show how to leverage the power of k-means and some interesting recent theory ([13, 35]) in order to better steer the partitioning decision at each iteration of PDDP. Our results confirm that such an approach leads to better performance compared to standard PDDP, in which the decision criteria are readily available from the spectral characteristics of the tdm.

Our second contribution is that we combine PDDP with kernel techniques, making the algorithm suitable for datasets not easily linearly separable. Specifically, we establish the necessary algebraic framework and propose KPDDP, a kernel version of PDDP as well as variants that we analyze and demonstrate that they return results of high quality compared to kernel k-means, albeit at higher cost.

The paper is organized as follows. Section 2 describes algorithms that generalize PDDP and introduces the concept of k-means steering. Section 3 reviews kernel PCA and Section 4 discusses the kernel versions of the above algorithms. Finally, Section 5 contains extensive experimental results.

As is frequently done, we will be using capital letters to denote matrices, lower case letters for (column) vectors and lower case Greek letters for scalars. When referring to specific matrix elements (scalars), we will use the lower case Greek letter corresponding to the Latin capital letter. When referring to vectors (rows or columns) of a matrix, we will use the corresponding lower case Latin letter. We also use diag($\cdot$) to denote the diagonal of its argument.

2 PDDP multi-partitioning and steering

It has long been known that the effectiveness of clustering algorithms depends on the application and data at hand. We focus on PDDP as an example of a spectral clustering algorithm that has been reported to be quite effective for high dimensional data, such as text, where standard density-based algorithms, such as DBSCAN, are not as effective because of their high computational cost and difficulty to achieve effective discrimination [6, 26]. PDDP, on the other hand, extracts spectral information from the data and uses it to constructs clusterings that can be of better quality than those provided by k-means. PDDP can be quite efficient in comparison to other agglomerative hierarchical

algorithms and can be used either as an independent clustering tool or as a "preconditioning tool" for the initialization of k-means.

We next highlight some features of PDDP that motivate the proposals and variants of our paper. At each step of PDDP, a cluster is selected and then partitioned in two subclusters using information from the PCA of the matrix corresponding to the data points belonging to the cluster. In this manner, the algorithm constructs a binary tree of clusters. More precisely, let p indicate the cluster node selected for partitioning at step i_p, $A^{(p)}$ the corresponding matrix (i.e. a submatrix of A with $n^{(p)}$ columns), $c^{(p)}$ its column centroid and $C^{(p)} = (A^{(p)} - c^{(p)}e^\top)(A^{(p)} - c^{(p)}e^\top)^\top$ the corresponding covariance matrix. During the split process, ordinary PDDP uses the first principal component of $A^{(p)}$. Equivalently, if $[u^{(p_i)}, \sigma^{(p_i)}, v^{(p_i)}]$ is the i-th singular triplet of the centered matrix $B^{(p)} = (A^{(p)} - c^{(p)}e^\top)$, the two subclusters are determined by the sign of the projection coefficients of each centered data point into the first principal component, $(u^{(p_1)})^\top(a^{(p_j)} - c^{(p)}) = \sigma^{(p_1)}v^{(p_1 j)}$, i.e. the sign of the corresponding element of $v^{(p_1)}$, since $\sigma^{(p_1)} > 0$.

Geometrically, the points $(u^{(p_1)})^\top(z - c^{(p)}) = 0$ define a hyperplane passing through the origin and perpendicular to $u^{(p_1)}$. Each data element is classified to one of two hyperplane-defined half-spaces. This can be extended to an 2^l-way partitioning strategy, based on information readily available from $l \geq 1$ singular vectors [33]. This method, named PDDP(l), classifies data points to one of 2^l orthants, based on the specific sign combination of the vector $(U_l^{(p)})^\top(a^{(p_j)} - c^{(p)}) = \Sigma_l^{(p)}(V_l^{(p)}(j,:))^\top$, where $U_l^{(p)}\Sigma_l^{(p)}(V_l^p)^\top$ the l-factor truncated SVD of $B^{(p)}$. Each one of $(u^{(p_i)})^\top(z - c^{(p)}) = 0$ defines a hyperplane passing through the origin orthogonal to $u^{(p_i)}$. In the sequel it would be useful to note that PDDP(l) reduces to the original PDDP algorithm when $l = 1$. Moreover, l is the dimension of the space in which lie the projection coefficients used to decide upon the splitting at each step of the algorithm. Finally, since PDDP(l) immediately classifies data into 2^l clusters, it can be used to partition $k = 2^l$ clusters in one single step.

Note that throughout our discussion, the cluster selected for partitioning will be the one with maximum scatter value, defined as $s_p = \|A^{(p)} - c^{(p)}e^\top\|_F^2$. This quantity is a measure of coherence, and it is the same as the one used by k-means [17, 23]. For convenience, we tabulate PDDP(l) in Table 1.

We have seen that with some work, PDDP classifies data from the cluster under consideration (possibly the entire dataset) into 2^l subclusters at the end of the first iteration. As outlined above, data are classified based on their orthant address, obtained from the corresponding sign combination of the l right singular vectors of the tdm. As we will see, there are cases where the sign combination is not a good cluster predictor and alternative classification techniques must be sought.

Algorithm PDDP(l)
Input: Feature×object matrix A (tdm), desired number of clusters k, number of principal components l
Output: Structure of `pddp_tree`, with $\left\lceil \frac{k-1}{2^l-1} \right\rceil (2^l - 1) + 1$ leaves
Initialize `pddp_tree`(A);
for $i = 1 : \left\lceil \frac{k-1}{2^l-1} \right\rceil$
  Select a cluster node p to split with $n^{(p)}$ elements;
  Let $A^{(p)}$ be the tdm of cluster node p, $c^{(p)}$ the corresponding centroid;
  Compute the leading l singular triplets $[u^{(p_j)}, s^{(p_j)}, v^{(p_j)}]$, $j = 1, ..., l$ of $(A^{(p)} - c^{(p)}e^\top)$;
  Assign each column $a^{(p_j)}$ of $A^{(p)}$ to node $2 + (i-2)2^l + j$, $j = 0 : 2^l - 1$ according to the signs of row j of $[v^{(p_1)}, ..., v^{(p_l)}]$;
  Update `pddp_tree`;
end

Table 1. Algorithm PDDP(l).

k-means steered PDDP

We next show how k-means can provide an effective steering mechanism for the partitioning at each step of PDDP(l). As we will see there are several alternatives; we first describe them in the context of the simplest version of PDDP. There, cluster membership of each datapoint is based on the sign of the corresponding element of the leading ($l = 1$) right singular vector of the centered tdm. Restating slightly, membership is decided based on whether the aforementioned element of the singular vector is smaller or larger than some "cut-point", which in this case is selected to be zero.

We call "cluster indicator vector problem" the computation of an indicator vector d with elements $\delta_i \in \{\pm 1\}$, whose values could imply a specific cluster assignment, e.g. assigning data element i to cluster C_1 (resp. C_2) if $\delta_i = 1$ (resp. "-1"). We make use of the following result:

Theorem 1 ([13]). *Let $A \in \mathbb{R}^{m \times n}$ be the tdm. For k-means clustering where $k = 2$, the continuous solution of the cluster indicator vector problem is the leading principal component, $r = [\rho_1, ..., \rho_n]^\top$, of A, i.e. data will be assigned to each of the two clusters according to the following index partitioning: $C_1 = \{i | \rho_i \leq 0\}, C_2 = \{i | \rho_i > 0\}$, where $i = 1, ..., n$.*

The above theorem suggests that if we relax the condition that $\delta_i \in \{\pm 1\}$ and allow the elements of the indicator vector to take any real value, the resulting vector would be r. Therefore, one partitioning technique could be to classify every data element in the cluster that is candidate for splitting based on the position of the corresponding element of r relative to 0. Theorem 1 suggests that splitting the data vectors according to r could be a good initialization for bisecting k-means. This technique can also be viewed as a

refinement of the basic PDDP, in which 2-means is applied to ameliorate the initial splitting. We refer to this approach as PDDP_2MEANS.

Initial centroids of 2-means are given by the ordering of r, i.e. $c_1 = (1/n_1)\sum_{i:\rho_i \leq 0} a_i$ and $c_2 = (1/n_2)\sum_{i:\rho_i > 0} a_i$ where n_1, n_2 denote respectively the numbers of negative or zero and positive elements of r. Therefore, the nondeterminism in (bisecting) k-means is eliminated.

We next note that even though r gives the solution of the relaxed problem, zero might not be the best cut-point. Indeed, Theorem 1 provides only an ordering for the data vectors, but not necessarily the best solution for the relaxed problem. In particular, another cut-point could lead to a smaller value for the k-means objective function. Specifically, we can check all possible splittings using the ordering implied by r, and select the one that optimizes the k-means objective function.

Lemma 1. *Let $A \in \mathbb{R}^{m\times n}$ and let $(j_1, j_2, \ldots, j_n)$ be an ordering of its columns. Then the optimal cut-point for 2-means can be obtained at cost $O(mn)$.*

Proof. Let $\{a_{j_1} a_{j_2} \ldots a_{j_n}\}$ the ordering of A's columns and $C_1 = \{a_{j_1} \ldots a_{j_l}\}$, $C_2 = \{a_{j_{l+1}} \ldots a_{j_n}\}$ a partition. The objective function value of C_1, C_2 is:

$$s_1 = \sum_{i=j_1}^{j_l} \|a_i - c_1\|^2 = \sum_{i=j_1}^{j_l} \|a_i\|^2 - l\|c_1\|^2$$

$$s_2 = \sum_{i=j_{l+1}}^{j_n} \|a_i - c_2\|^2 = \sum_{i=j_{l+1}}^{j_n} \|a_i\|^2 - (n-l)\|c_2\|^2$$

while the ojective function value for the clustering is given by $s = s_1 + s_2$. Consider now the partition $C_1 = \{a_{j_1} \ldots a_{j_l} a_{j_{l+1}}\}$, $C_2 = \{a_{j_{l+2}} \ldots a_{j_n}\}$. The new centroid vectors are given by:

$$\hat{c}_1 = \frac{lc_1 + a_{j_{l+1}}}{l+1}, \hat{c}_2 = \frac{lc_2 - a_{j_{l+1}}}{n-l-1}$$

while:

$$\hat{s} = \sum_{i=j_1}^{j_n} \|a_i\|^2 - (l+1)\hat{c}_1 - (n-l-1)\hat{c}_2$$

Clearly, the update of the centroid vectors requires $O(m)$ operations as well the computation of the new value of the objective function. This operation is applied $n-1$ times and the proof follows. □

Based on the above proof, we can construct an algorithm to determine the optimal cut-point corresponding to the elements of the leading principal component of A. The resulting splitting can be used directly in PDDP; alternatively, it provides a starting point for the application of 2-means. We refer to the algorithms corresponding to these two options as PDDP_OC and

PDDP_OC_2MEANS respectively. The above techniques can be extended to the case of PDDP(l).

We next note that in the course of their 2-means phase, all the above enhancements of PDDP (PDDP_2MEANS, PDDP_OC, PDDP_OC_2MEANS) necessitate further operations between m-dimensional data vectors. Despite the savings implied by Lemma 1, costs can quickly become prohibitive for datasets of very high-dimensionality. On the other extreme, costs would be minimal for datasets consisting of only one feature. We can thus consider constructing algorithms in which we apply the above enhancements, not on the original dataset but on some compressed representation. We know, for example, that in terms of the Euclidean norm, the optimal unit rank approximation of the centered tdm can be written as $\sigma_1 u_1 v_1^\top$, where $\{\sigma_1, u_1, v_1\}$ is its largest singular triplet. Vector v_1 contains the coefficients of the projection of the centered tdm on the space spanned by u_1 and can be used a one-dimensional encoding of the dataset. We further refine the usual PDDP policy, which utilizes zero as cut-point, i.e. employs a sign-based splitting induced by this vector. In this manner, however, the magnitude of the corresponding elements plays no role in the decision process.

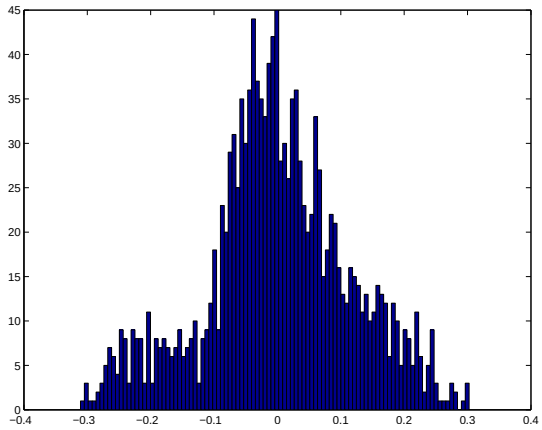

Fig. 1. Histogram of the projection coefficients employed by PDDP.

To see that this could lead to an inferior partitioning, we depict in Fig. 1 a histogram of the elements of v_1 corresponding to part of the Reuters-21578 collection. Many elements cluster around the origin, however a value near -0.1 appears to be a more reasonable choice for cut-point than 0. Our next proposal is an algorithm that addresses this issue in the spirit of the previous enhancements of PDDP. Specifically, we propose the application of the 2-means algorithm on the elements of v_1. A key feature of this proposal is the fact that 2-means is applied on a one-dimensional dataset. We can thus evade NP-hardness, in fact we can compute the optimal cut-point so that the

centroids of the two subsets of data points on either side of this point maximize the objective function of 2-means. To accomplish this, it is sufficient to sort the elements of v_1 and examine all possible partitions. Since the objective function for each attained partitioning can be computed in constant time and there are only $n-1$ possible partitions, the entire process can be completed at a cost of only $O(n)$ operations. Note that this is a direct extension of Lemma 1 for the case of one-dimensional datasets.

We can apply the same principle to PDDP(l), where $l > 1$, by identifying the optimal cut-point along each one of the l leading principal components (equivalently, the leading right singular vectors). If $z := [\zeta_1 \ \ldots \ \zeta_l]^\top$ consists of the l cut-points that are optimal for each dimension, we can apply the basic PDDP(l) procedure by first shifting the origin by z. We denote by PDDP_OCPC the implied technique. Note that all the above policies eliminate the random start from 2-means and make it deterministic.

It is fair to mention here that the potential for alternative partitioning strategies was already mentioned in the original PDDP paper[5]. Two related approaches are [11] and [19]. The former paper proposes the use of k-means to best assign scalar data into k clusters, the case $k = 2$ being the most prominent. In the latter, an algorithm called spherical PDDP (sPDDP for short) is proposed, employing $l = 2$ principal components during the split procedure and computing the optimal separation line into the circle defined by the normalized vectors of the projection coefficients.

We finally note that PDDP could be deployed as a preconditioner, to determine the k initial clusters necessary to start the iterations of k-means. This idea was described in [31] and also resolves the indeterminacy in k-means but is fundamentally different from our proposed k-means steered PDDP variants.

3 Kernel PCA

The key idea in kernel learning methods is the mapping of a set of data points into a more informative high dimensional feature space through a nonlinear mapping. Using Mercer kernels, any algorithm that can be expressed solely in terms of inner products, can be applied without computing explicitly the mapping. Kernel methods have been used widely in support vector machine (SVM) research [10]. In the context of clustering, Finley and Joachims in [15] and Ben-Hur et al. in [3] propose some SVM based clustering algorithms. Schölkopf et al. in [29] and Camastra and Verri in [9] propose the kernel k-means algorithm, while Zhang and Chen in [36] propose a kernel fuzzy c-means variant. Dhillon et al. in [12] provide a useful connection between kernel k-means and spectral clustering.

[5] As stated in [7] "... the values of the singular vector are used to determine the splitting ... in the simplest version of the algorithm we split the documents strictly according to the sign ..."

PCA is an important statistical tool used in a wide range of applications where there is a need for dimensionality reduction. Kernel PCA (KPCA) is an extension of PCA that is applied to a vector space defined by a nonlinear mapping and has been used with success in a wide range of applications (e.g. [20, 24, 28, 29, 32]). In particular, let ϕ be a general mapping, $\phi : X \to F$, mapping input vectors $x \in X$ to vectors $\phi(x) \in F$. We assume that X is a subset of $\mathbb{R}^m$. KPCA amounts to the application of standard PCA to a new vector space, called feature space. Let $\langle \cdot, \cdot \rangle$ denote the Euclidean inner product. Inner products in F can be computed via the "kernel trick", that is by defining an appropriate symmetric kernel function, $k(x, y)$ over $X \times X$, which satisfies Mercer's theorem ([10]) so that

$$k(x^{(i)}, x^{(j)}) := \langle \phi(x^{(i)}), \phi(x^{(j)}) \rangle \tag{1}$$

for $x^{(i)}, x^{(j)} \in X$. There are various kernel functions, with the polynomial, $k(x, y) = (x^\top y + 1)^d$ and Gaussian, $k(x, y) = \exp(-\|x - y\|^2/\sigma^2)$, being two of the most common. It then becomes possible to express algorithms over F that use only inner products via kernels, specifically without ever computing the $\phi(x^{(i)})$'s.

Assume for now that the $\phi(x^{(i)})$'s are already centered. The covariance matrix of $\tilde{A} = [\phi(x^{(1)}) \ \dots \ \phi(x^{(n)})]$ is

$$\tilde{C} = \sum_{i=1}^{n} \phi(x^{(i)}) \phi(x^{(i)})^\top = \tilde{A}\tilde{A}^\top.$$

The principal components of $\tilde{A}$ are the eigenvectors of $\tilde{C}$, that is the solutions of $\tilde{C}v = \lambda v$. Each v can be expressed as a linear combination of the columns of $\tilde{A}$, i.e. $v = \sum_{i=1}^{n} \psi_i \phi(x^{(i)})$. The latter can be rewritten as $v = \tilde{A}y$, where $y = [\psi_1, ..., \psi_n]^\top$, therefore projecting the eigenvalue problem into each column of $\tilde{A}$, it turns out that the PCA of $\tilde{A}$ amounts to an eigenvalue problem for the Gram matrix, namely solutions of $Ky = \lambda y$ where $(K)_{i,j} := \kappa_{ij} = \langle \phi(x^{(i)}), \phi(x^{(j)}) \rangle$. This can be solved using (1) without forming the $\phi(x^{(i)})$'s, assuming that the Gram matrix K is available. The last step is the normalization of the eigenvectors $y^{(j)}, j = 1, \dots, n$. This is necessary since $\langle v^{(j)}, v^{(j)} \rangle = 1$ must hold:

$$\begin{aligned} 1 = \langle v^{(j)}, v^{(j)} \rangle &= \langle \sum_{i=1}^{n} \psi_i^{(j)} \phi(x^{(i)}), \sum_{r=1}^{n} \psi_r^{(j)} \phi(x^{(r)}) \rangle \\ &= \sum_{i,r=1}^{n} y_i^{(j)} \psi_r^{(j)} \langle \phi(x^{(i)}), \phi(x^{(r)}) \rangle \\ &= \langle y^{(j)}, K y^{(j)} \rangle = \lambda_j \langle y^{(j)}, y^{(j)} \rangle = \lambda_j \|y^{(j)}\|_2^2 \end{aligned}$$

So, the final step is the normalization of $y^{(j)}$ according to

$$\|y^{(j)}\|_2 = \frac{1}{\sqrt{\lambda_j}} \tag{2}$$

The coefficients of the projection of a column $\phi(x^{(r)})$ of $\tilde{A}$ into an eigenvector $v^{(j)}$ of $\tilde{C}$ are given by

$$\langle v^{(j)}, \phi(x^{(r)})\rangle = \sum_{i=1}^{n} y_i^{(j)} \langle \phi(x^{(i)}), \phi(x^{(r)})\rangle = (y^{(j)})^\top K_{:,r}$$

where $K_{:,r}$ denotes the r-th column of the Gram matrix.

Setting Y_k to be the matrix formed by the k leading eigenvectors of K, $Y_k^\top K$ contains a k-dimensional representation for each vector $\phi(x^{(j)})$. If the $\phi(x^{(i)})$'s are not centered, it was shown in [29] that the principal components of $\tilde{C}$ are given by the eigenvectors of

$$\tilde{K} = K - \frac{1}{n} ee^\top K - \frac{1}{n} Kee^\top + \frac{1}{n^2} ee^\top Kee^\top \tag{3}$$

After some algebraic manipulation, $\tilde{K}$ can be expressed as $(I-P)K(I-P)$, where $P := ee^\top/n$ is an orthogonal projector. In this case the projection coefficients of the data points into the first k principal components are $Y_k^\top \tilde{K}$, where Y_k consists of the leading k eigenvectors of $\tilde{K}$.

4 Kernel Clustering

Our proposal is to use KPCA in place of PCA during the splitting step of PDDP. Assuming that, based on scatter, p is the cluster node selected for partitioning at step i_p, we accomplish this using information from the KPCA of the corresponding matrix $A^{(p)}$, i.e. from the linear PCA of $\hat{B}^{(p)} = \Phi(B^{(p)}) = \Phi(A^{(p)} - c^{(p)}e^\top)$, where $\Phi(\cdot)$ denotes the matrix $\Phi(X)^{(j)} = \phi(x^{(j)})$ and $\phi(\cdot)$ a nonlinear mapping (e.g. polynomial kernel).

As indicated in Eq. (3), the principal components of $\hat{B}^{(p)}$ are given by the eigenvectors of

$$\hat{K}^{(p)} = K^{(p)} - MK^{(p)} - K^{(p)}M + MK^{(p)}M \tag{4}$$

where $M = \frac{1}{n^{(p)}} ee^\top$ and $[K_{ij}^{(p)}] = [\langle \phi(a^{(p_i)} - c^{(p)}), \phi(a^{(p_j)} - c^{(p)})\rangle]$ the Gram matrix corresponding to cluster p. The projection coefficients of $\hat{B}^{(p)}$ into the top l eigenvectors of the covariance matrix $(\hat{B}^{(p)})(\hat{B}^{(p)})^\top$ are given by:

$$(V_l^{(p)})^\top \hat{B}^{(p)} \in \mathbb{R}^{l \times n^{(p)}} \tag{5}$$

where $V_l^{(p)}$ are the l leading eigenvectors of $[K_{ij}^{(p)}]$. Note that the normalization condition for the eigenvectors of $\hat{K}^{(p)}$ also causes the normalization of $V^{(p)}$'s columns according to (2).

Regarding the cluster selection step, note that:

Algorithm KPDDP(l)
Input: Feature×object matrix A (tdm), desired number of clusters k, number of principal components l
Output: Structure of `pddp_tree`, with $\left\lceil \frac{k-1}{2^l-1} \right\rceil (2^l-1)+1$ leaves
Initialize `pddp_tree`(A);
for $i = 1 : \left\lceil \frac{k-1}{2^l-1} \right\rceil$
  Select a cluster node p to split according to (6); Let $A^{(p)}$ be the tdm of cluster p, $c^{(p)}$ the corresponding centroid;
  Form the Gram matrix $K^{(p)}$;
  Compute the leading l eigenvectors and eigenvalues of $\hat{K}^{(p)}$;
  Normalize the eigenvectors $V^{(p_i)}$, $i = 1, \ldots, l$ of $K^{(p)}$ according to (2);
  Assign each column $a^{(p_j)}$ of $A^{(p)}$ to cluster node $2 + (i-2)2^l + j$, $j = 0 : 2^l - 1$ according to the signs of row j of $(V_l^{(p)})^\top \hat{B}^{(p)}$;
  Update `pddp_tree`;
end

Table 2. Algorithm KPDDP(l).

$$
\begin{aligned}
s_{p_\phi} &= \|\Phi(A^{(p)}) - \hat{c}^{(p)}(e^{(p)})^\top\|_F^2 \\
&= \sum_{j=1}^{n^{(p)}} \|\hat{a}^{(p_j)} - \hat{c}^{(p)}\|_F^2 = \sum_{j=1}^{n^{(p)}} \|\hat{a}^{(p_j)} - \hat{c}^{(p)}\|_2^2
\end{aligned}
$$

where $\hat{c}^{(p)}$ the centroid of $\Phi(A^{(p)})$ into the feature space defined by $\phi(\cdot)$. Since $\|x-y\|_2^2 = \langle x,x\rangle^2 + \langle y,y\rangle^2 - 2\langle x,y\rangle$, it follows that the scatter value can be computed from

$$
\begin{aligned}
s_{p_\phi} &= \sum_{j=1}^{n^{(p)}} \|\phi(a^{(p_j)})\|_2^2 + \sum_{j=1}^{n^{(p)}} \|\hat{c}^{(p)}\|_2^2 - 2\sum_{j=1}^{n^{(p)}} \langle \phi(a^{(p_j)}), \hat{c}^{(p)} \rangle \\
&= \sum_{j=1}^{n^{(p)}} \langle \phi(a^{(p_j)}), \phi(a^{(p_j)}) \rangle + n^{(p)} \langle \frac{\Phi(A^{(p)})e}{n^{(p)}}, \frac{\Phi(A^{(p)})e}{n^{(p)}} \rangle \\
&\quad -2\sum \langle \phi(a^{(p_j)}), \Phi(A^{(p)})e^{(p)} \rangle \\
&= \sum_{j=1}^{n^{(p)}} \langle \phi(a^{(p_j)}), \phi(a^{(p_j)}) \rangle + \frac{\|\bar{K}^{(p)}e\|_1}{n^{(p)}} - 2\frac{\|\bar{K}^{(p)}e\|_1}{n^{(p)}}
\end{aligned}
$$

where $\bar{K}_{ij}^{(p)} = \langle \phi(a^{(p_i)}), \phi(a^{(p_j)}) \rangle$ denotes the Gram matrix of $A^{(p)}$. Therefore,

$$
s_{p_\phi} = \mathrm{trace}(\bar{K}^{(p)}) - 2\frac{\|\bar{K}^{(p)}e\|_1}{n^{(p)}} \tag{6}
$$

We call the proposed algorithm KPDDP(l) (Kernel PDDP(l)). The algorithm is tabulated in Table 2. As in the basic algorithm, at each step the cluster node p with the largest scatter value (6) is selected and partitioned into 2^l subclusters. Each member of the selected node is classified into the subcluster defined by the combination of its projection coefficients (5) into the l principal components of $\hat{B}^{(p)}$. In our implementation, we experimented with polynomial and Gaussian kernels. Our formulation, however, is general and can be applied for any kernel. The leading eigenpairs of the covariance matrix were computed from the SVD of the centered tdm using PROPACK [21]. This consists of a very efficient MATLAB implementation that applies Lanczos bidiagonalization with partial reorthogonalization to compute selected singular triplets. The algorithm requires access to a routine to form matrix-vector products $K^{(p)}x$, but does not necessitate the explicit construction of the Gram matrix $\hat{K}^{(p)}$.

Kernel *k*-means

Kernel learning has already been applied in k-means clustering; e.g. [9, 29] outline efficient implementations. As in the linear case, each cluster is represented by its centroid into the new feature space, $\hat{c}^{(p)} = \frac{1}{n_p}(\phi(a_{p_1}) + \ldots + \phi(a_{p_n}))$, however there is no need to compute $\hat{c}^{(p)}$'s. In particular, the algorithm operates iteratively as k-means by assigning each data point to the nearest cluster and updates centroids using the last assignment. The norm of the Euclidean distance of x, y equals to $\|x-y\|_2^2 = \langle x,x\rangle + \langle y,y\rangle - 2\langle x,y\rangle$. Denoting by K the Gram matrix of A and using MATLAB notation, the distance of a single data point a_j from a centroid $\hat{c}^{(p)}$ is $\langle a_j, a_j\rangle + \texttt{diag}(K)_{p_1\ \ldots\ p_n} - 2K_{j,(p_1\ \ldots\ p_n)}$. $\langle a_j, a_j\rangle$'s can be computed initially, and then during each iteration the cluster membership indication matrix can be computed as $P = ex^\top + ye^\top - 2Z$, where x contains the norms of A's columns, y the norms of centroids computed from a partial sum of K's elements once in each itaration and Z the $k \times n$ matrix with elements $\zeta_{i,k} = \langle \hat{c}^{(i)}, a_j\rangle$. The algorithm assigns each element j to cluster $i_j = \arg\{\min_i P(i,j)\}$.

k-means can be extended as in the linear case in order to get a hierarchical clustering solution (kernel bisecting k-means). Furthermore, using the same reasoning, we can combine the kernel versions of PDDP and k-means in order to derive more effective clustering solutions. In particular, we propose the use of the kernel 2-means algorithm during the splitting process of KPDDP (we refer to it as KPDDP_2MEANS). Finally, we can extend the PDDP_OCPC variant in the kernel learning framework[6].

[6] We note that methods PDDP_OC and PDDP_OC_2MEANS can also be extended in kernel variants (see Theorem 3.5 in [13]), however we observed that this results in a high computational overhead that makes these approaches prohibitive.

5 Experimental Results

feature	MODAPTE	OHSUMED	CLASSIC3	REUT1	REUT2	REUT3	REUT4
documents	9,052	3,672	3,891	840	1,000	1,200	3,034
terms	10,123	6,646	7,823	2,955	3,334	3,470	5,843
terms/document	60	81	77	76	75	60	78
tdm nonzeros (%)	0.37	0.76	0.64	1.60	1.43	0.37	84
number of clusters	52	63	3	21	10	6	25

Table 3. Dataset Statistics.

We next conduct extensive experiments in order to evaluate the impact of the proposed techniques. For this purpose, we use the well known Reuters-21578, Ohsumed (part of the TREC filtering track) and CLASSIC3 (a merge of MEDLINE, CRANFIELD and CISI) collections. For the linear case, we use the ModApte split of the Reuters-21578 collection as well as part of the Ohsumed collection composed of those documents which belong to a single cluster. Furthermore, we use CLASSIC3 for a toy example. For the nonlinear case, we use 4 datasets, named REUTj ($j = 1, ..., 4$), constructed from the ModApte split with varying number of clusters and cluster sizes[7]. Table 3 depicts the characteristics of those collections. TMG [34] has been used for the construction of the tdm's. Based on [34], we used logarithmic local term and IDF global weightings with normalization, stemming and stopword removal, removing also terms that appeared only once in the collection. Our experiments were conducted on a Pentium IV PC with 1GB RAM using MATLAB.

In the following discussion, we denote by k the sought number of clusters. For each dataset we ran all algorithms for a range of k. In particular, denoting by r the true number of clusters for a dataset, we ran all algorithms for $k = 4 : 3 : k_{\max}$ and $k = 8 : 7 : k_{\max}$ for some $k_{\max} > r$ in order to record the results of PDDP(l) and related variants for $l = 1, 2, 3$. For all k-means variants we have conducted 10 experiments with random initialization of centroids and recorded the minimum, maximum and mean values of attained accuracy and runtime. Although we present only mean-value results, minimum and maximum values are improrant to the discussion that follows. For the SVD and eigendecomposition we used the MATLAB interface of the PROPACK software package [21]. For the algorithms' evaluation, we use the objective function of k-means (and PDDP), the entopy and runtime measures.

Fig. 2 depicts the objective function, entropy values and runtime for all variants, for the linear case and datasets MODAPTE and OHSUMED. Although k-means appears to give the best results between all variants and all measures, we note that these plots report mean values attained by k-means and related variants. In practice, a single run of k-means may lead to poor results. As a

[7] We will call the ModApte and Ohsumed datasets as MODAPTE, OHSUMED.

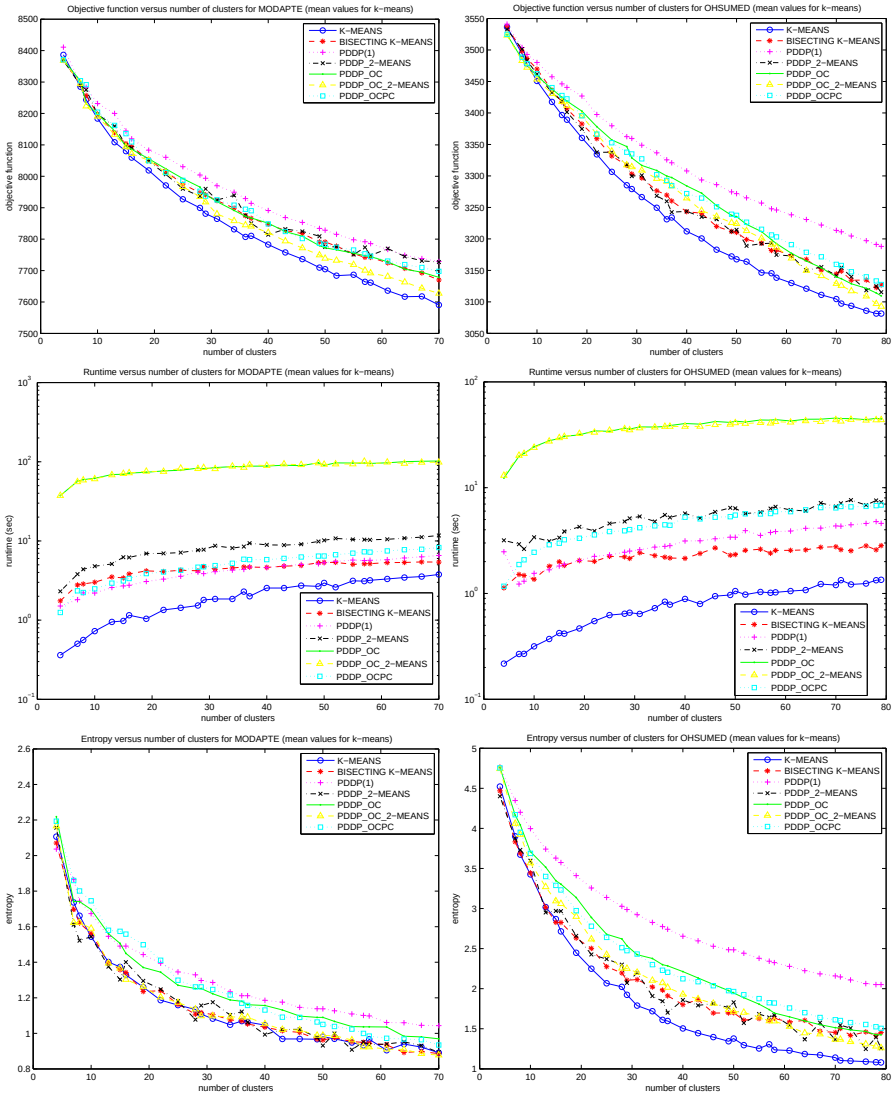

Fig. 2. Objective function, entropy values and runtime for k-means, PDDP and variants.

result, a "good" partitioning may require several executions of the algorithm. Compared to the basic algorithm, its hierarchical counterpart (bisecting k-means) appears to degrade the quality of clustering and suffers from the same problems as k-means. On the other hand, PDDP appears to give results inferior to k-means. Regarding the proposed variants, we note that all techniques always improve PDDP and bisecting k-means in most cases, while approach-

ing the clustering quality of k-means. PDDP_OC_2MEANS and PDDP_OC appear to give the best results, however their high runtime make them relative expensive solutions. On the other hand, PDDP_2MEANS and PDDP_OCPC provide results similar to k-means without impacting significantly the overall efficiency of PDDP.

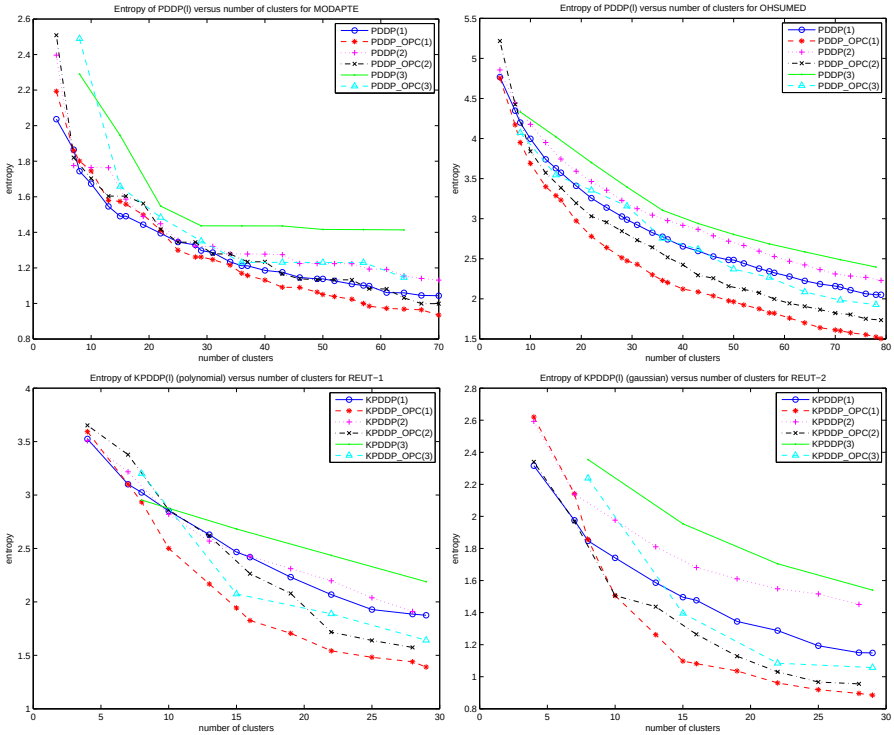

Fig. 3. Objective function and entropy values for PDDP and PDDP_OCPC and $l = 1, 2, 3$, linear (upper) and nonlinear (lower) case.

Fig. 3 (upper) depicts the quality of clustering of PDDP_OCPC vs. PDDP(l) for $l = 1, 2, 3$. PDDP_OCPC appears to improve significantly the efficiency of PDDP(l) for equal values of parameter l. A simple example that demonstrates the success of PDDP_OCPC is given in Table 4, where we give the confusion matrices for PDDP and PDDP_OCPC for CLASSIC3 and $k = 3, 4$. PDDP_OCPC appears to approximate better the cluster structure of the collection by producing "cleaner" clusters.

In Fig. 4, 5 (polynomial and gaussian kernels respectively) we give the results for the kernel versions of the algorithms for datasets REUT1-REUT4. As in the linear case, kernel k-means appears to give better results than kernel PDDP. However, our hybrid schemes appear to improve even the k-means

cluster	1	2	3	1	2	3	4
class 1	12	6	1,015	12	1,015	4	2
class 2	1,364	14	20	1,364	20	8	6
class 3	2	1,392	66	2	66	788	604
class 1	0	9	1,024	0	1,024	7	2
class 2	1,253	29	116	1,253	116	23	6
class 3	0	1,431	29	0	29	917	514

Table 4. Confusion matrices for CLASSIC3 for PDDP (upper left quadrant for $k = 3$ and right for $k = 4$) and PDDP_OCPC (lower left quadrant for $k = 3$ and right for $k = 4$).

algorithm at low additional runtime. Fig. 3 (lower) depicts the quality of clustering of the kernel versions PDDP_OCPC vs. PDDP(l) for $l = 1, 2, 3$. As in the linear case, PDDP_OCPC appears to provide significant improvements over PDDP. It is worth noting that in our experiments, results with the linear versions appeared to be uniformly better than results with the kernel implementations. This does not cause concern since our goal was to improve kernel k-means algorithm along deterministic approaches that are expected to give better results in case there are strong nonlinearities in data at hand.

The above results indicate that our hybrid clustering methods that combine k-means and PDDP can be quite successful in addressing the nondeterminism in k-means, while achieving at least its "average-case" effectiveness. The selection of a specific technique can be dictated by the quality or runtime constraints imposed by the problem. Furthermore, we proposed a kernel version of the PDDP algorithm, along some variants that appear to improve kernel version of both PDDP and k-means. The implications on memory caused by the use of the tdm Gramian in kernel methods are currently under investigation.

Acknowledgements

An earlier version of this paper was presented at and included in the proceedings of the *Text Mining Workshop* held during the *2007 SIAM International Conference on Data Mining*. The Workshop was organized by Michael Berry and Malu Castellanos. We thank them and Murray Browne for inviting us to contribute to the current volume. We also thank Dan Boley and Jacob Kogan for dicussions on topics related to the subject of this paper. Part of this work was conducted when the authors were supported by a University of Patras K. Karatheodori grant (no. B120).

References

1. C.J. Alpert, A.B. Kahng, and S.-Z. Yao. Spectral partitioning with multiple eigenvectors. *Discrete Applied Mathematics*, 90:3–26, 1999.

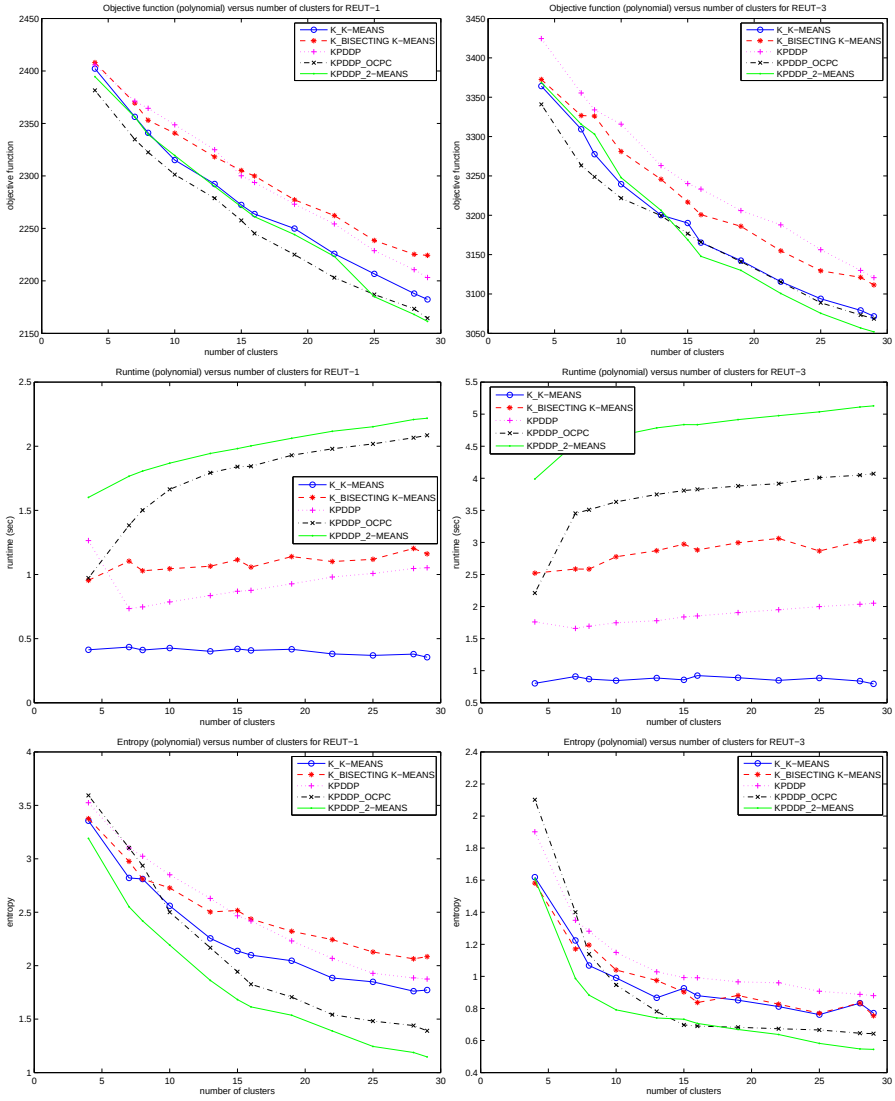

Fig. 4. Objective function, entropy values and runtime for kernel versions of k-means, PDDP and variants (polynomial kernels).

2. C.J. Alpert and S.-Z. Yao. Spectral partitioning: the more eigenvectors, the better. In *Proc. 32nd ACM/IEEE Design Automation Conf.*, pages 195–200. ACM Press, 1995.
3. A. Ben-Hur, D. Horn, H.T. Siegelmann, and V. Vapnik. Support vector clustering. *Machine Learning Research*, 2:125–137, 2001.
4. P. Berkhin. A survey of clustering data mining techniques. In C. Kogan, J. Nicholas and M. Teboulle, editors, *Grouping Multidimensional Data: Recent*

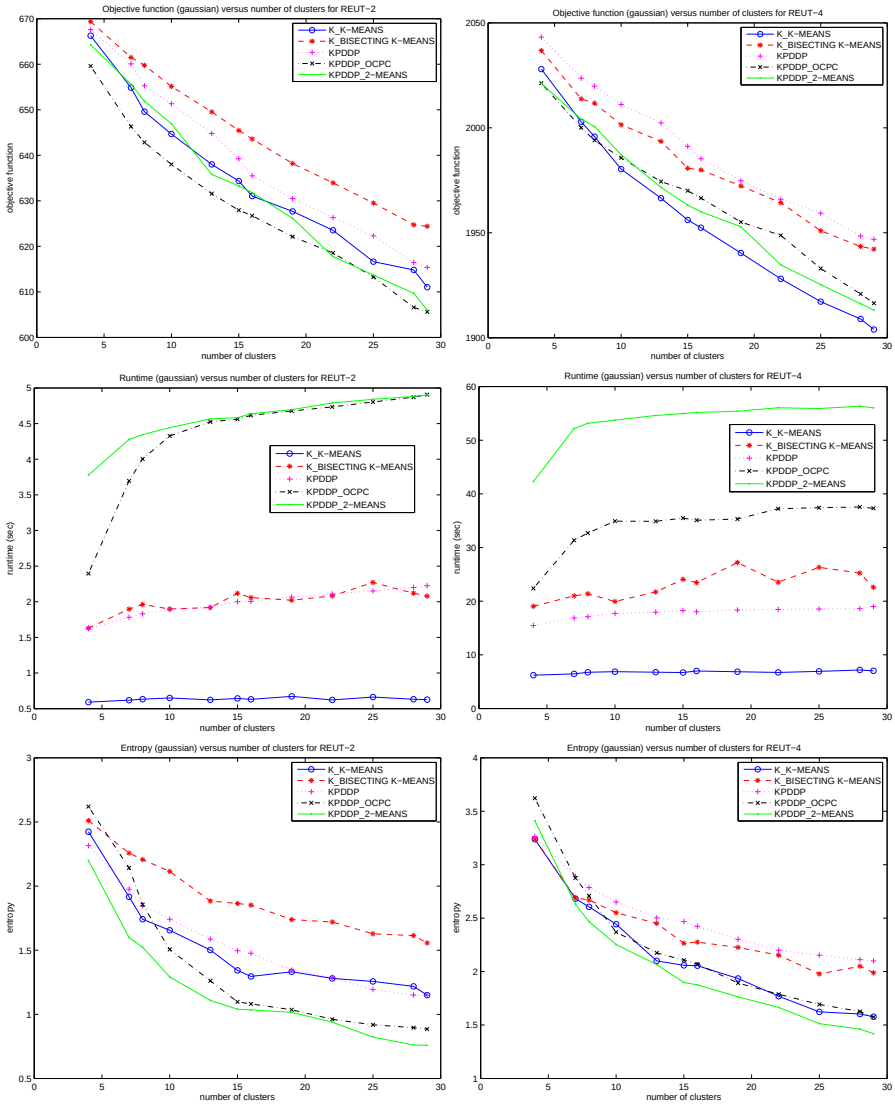

Fig. 5. Objective function, entropy values and runtime for kernel versions of k-means, PDDP and variants (gaussian kernel).

Advances in Clustering, pages 25–72. Springer, Berlin, 2006.

5. M.W. Berry. Large scale singular value decomposition. *Int'l. J. Supercomp. Appl.*, 6:13–49, 1992.
6. K. Beyer, J. Goldstein, R. Ramakrishnan, and U. Shaft. When is "nearest neighbor" meaningful? In *Lecture Notes in Computer Science*, volume 1540, pages 217–235. 1999.

7. D. Boley. Principal direction divisive partitioning. *Data Mining and Knowledge Discovery*, 2(4):325–344, 1998.
8. D. Boley. A scalable hierarchical algorithm for unsupervised clustering. In R. Grossman, C. Kamath, P. Kegelmeyer, V. Kumar, and R. Namburu, editors, *Data Mining for Scientific and Engineering Applications*. Kluwer Academic Publishers, 2001.
9. F. Camastra and A. Verri. A novel kernel method for clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(5):801–804, 2005.
10. N. Cristianini and J. Shawe-Taylor. *An Introduction to Support Vector Machines and other kernel-base learning methods*. Cambridge University Press, 2000.
11. I.S. Dhillon. Co-clustering documents and words using bipartite spectral graph partitioning. In *In Proc. 7th ACM SIGKDD*, pages 269–274, New York, 2001. ACM Press.
12. I.S. Dhillon, Y. Guan, and B. Kulis. Kernel k-means: spectral clustering and normalized cuts. In *Proc. 10th ACM SIGKDD*, pages 551–556, New York, 2004. ACM Press.
13. C. Ding and X. He. Cluster structure of k-means clustering via principal component analysis. In *PAKDD*, pages 414–418, 2004.
14. W.E. Donath and A.J. Hoffman. Lower bounds for the partitioning of graphs. *IBM J. Res. Develop.*, 17:420–425, 1973.
15. T. Finley and T. Joachims. Supervised clustering with support vector machines. In *In ICML '05: Proceedings of the 22nd international conference on Machine learning*, pages 217–224, New York, 2005.
16. B. Hendrickson and R. Leland. An improved spectral graph partitioning algorithm for mapping parallel computations. *SIAM J. Sci. Comput.*, 16(2):452–469, 1995.
17. T. Kanungo, D.M. Mount, N.S. Netanyahu, and A.Y. Platko, D. Wu. An efficient k-means clustering algorithm: Analysis and implementation. *IEEE Trans. PAMI*, 24(7):881–892, 2002.
18. J. Kogan. *Introduction to Clustering Large and High-Dimensional Data*. Cambridge University Press, 2007.
19. J. Kogan, I. S. Dhillon, and C. Nicholas. Feature selection and document clustering. In M. Berry, editor, *A Comprehensive Survey of Text Mining*. Springer, 2004.
20. E. Kokiopoulou and Y. Saad. PCA and kernel PCA using polynomial filtering: A case study on face recognition. In *In SIAM Conf. on Data Mining*, 2005.
21. R.M. Larsen. Propack: A software package for the symmetric eigenvalue problem and singular value problems on lanczos and lanczos bidiagonalization with partial reorthogonalization. Stanford University, http://sun.stanford.edu/∼rmunk/PROPACK/.
22. D. Littau and D. Boley. Clustering very large data sets with principal direction divisive partitioning. In C. Kogan, J. Nicholas and M. Teboulle, editors, *Grouping Multidimensional Data: Recent Advances in Clustering*, pages 99–126. Springer, Berlin, 2006.
23. S.P. Lloyd. Least squares quantization in PCM. *IEEE Trans. Information Theory*, 28:129–137, 1982.
24. K. R. Müller, S. Mika, G. Rätsch, K. Tsuda, and B. Schölkopf. An introduction to kernel-based learning algorithms. *IEEE Transactions on Neural Networks*, 12(2):181–202, 2001.

25. G. Salton and C. Buckley. Term weighting approaches in automatic text retrieval. *Information Processing and Management*, 24(5):513–523, 1988.
26. J. Sander, M. Ester, H.-P. Kriegel, and X. Xu. Density-based clustering in spatial databases: The algorithm GDBSCAN and its applications. *Data Mining and Knowledge Discovery*, 2(2):169–194, 1998.
27. S. Savaresi, D. Boley, S. Bittanti, and G. Gazzaniga. Choosing the cluster to split in bisecting divisive clustering algorithms. In *In Second SIAM International Conference on Data Mining (SDM'2002)*, 2002.
28. A. J. Schölkopf, B. Smola and K. R. Müller. Kernel principal component analysis. In *In ICANN*, pages 583–588, 1997.
29. B. Schölkopf, A. J. Smola, and K. R. Müller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation*, 10(5):1299–1319, 1998.
30. M. Steinbach, G. Karypis, and V. Kumar. A comparison of document clustering techniques, 2000. In KDD Workshop on Text Mining.
31. S. Xu and J. Zhang. A parallel hybrid Web document clustering algorithm and its performance study. *J. Supercomputing*, 30(2):117–131, 2004.
32. M. H. Yang, N. Ahuja, and D. J. Kriegman. Face recognition using kernel eigenfaces. In *In ICIP*, 2000.
33. D. Zeimpekis and E. Gallopoulos. PDDP(l): Towards a flexible principal direction divisive partitioning clustering algorithm. In D. Boley, I. Dhillon, J. Ghosh, and J. Kogan, editors, *Proc. Workshop on Clustering Large Data Sets (held in conjunction with the Third IEEE Int'l. Conf. Data Min.)*, pages 26–35, Melbourne, FL, Nov. 2003.
34. D. Zeimpekis and E. Gallopoulos. TMG: A MATLAB toolbox for generating term-document matrices from text collections. In J. Kogan, C. Nicholas, and M. Teboulle, editors, *Grouping Multidimensional Data: Recent Advances in Clustering*, pages 187–210. Springer, 2006.
35. H. Zha, X. He, C. Ding, M. Gu, and H. Simon. Spectral relaxation for k-means clustering. In *NIPS*, pages 1057–1064, 2001.
36. D. Q. Zhang and S. C. Chen. Clustering incomplete data using kernel-based fuzzy c-means algorithm. *Neural Processing Letters*, 18(3):155–162, 2003.

Index