THE K-DISCS ALGORITHM AND ITS KERNELIZATION

MICHAEL LINDSEY

Abstract. A new clustering algorithm called k-discs is introduced. This algorithm, though similar to k-means, addresses the deficiencies of k-means as well as its variant k-subspaces. The k-discs algorithm is applied to the recovery of manifolds from noisy samplings. A kernelization of k-discs is exhibited, and the advantages of this modification are demonstrated on synthetic data. For this project, the kernelized and linear algorithms were implemented from scratch in MATLAB.

1. Introduction and Motivation

A clustering algorithm is usually run on a data set under the assumption that there is some structure of various connected components underlying the data set (or to test whether this is a good assumption). Suppose that a set of n-dimensional data has been sampled, with noise, from a smooth submanifold (potentially with boundary) of $\mathbb{R}^n$. One wishes to recover, in some sense, the underlying structure of the data, given only the data itself. Arguably the most fundamental aspect of this structure is the characterization of the connected components of the manifold, manifested as clusters in the data set. Unfortunately, standard clustering algorithms such as k-means and expectation-maximization often fail to properly characterize the shapes of these components. Kernel k-means succeeds in many circumstances in clustering data that is not separable by ordinary linear k-means. However, due to its zero-dimensional characterization of clusters, it fails to identify clusters drawn from elongated subsets of submanifolds.

k-subspaces is a variant of k-means which seeks to minimize the total distance of the data from a set of k d-dimensional subspaces. The kernelization of k-subspaces achieves good results in identifying clusters drawn from nonlinear d-dimensional submanifolds (in particular, submanifolds without boundary). However, the infinite extent of this algorithm's clusters can result in awkward clustering. For example, k-subspaces will have difficulty in identifying bounded clusters drawn from the same subspace (or the same submanifold).

2. The k-discs Algorithm

In short, the k-subspaces algorithm attempts to minimize the total distance of the data from a set of k d-dimensional discs. Since the extent of data is finite, we do not pay a great price for viewing clusters as bounded subsets of subspaces. In fact, this algorithm can be seen as a refinement of both k-means and k-subspaces in that it:

- can identify each point with a cluster centroid (the center of its assigned disc) or with its projection onto the subspace containing the disc
- endows clusters with some (non-zero-dimensional) notion of shape and scalar measure
- allows for multiple clusters within or near the same subspace.

3. Algorithmic Details

The k-discs algorithm consists of the following steps. All points are in $\mathbb{R}^n$, and we are given a data set $\{x^{(1)}, \dots, x^{(m)}\}$.

(1) For a given k (number of clusters) and d (dimension of cluster discs), initialize k cluster centroids μ_j and a random set $\{u_{jl}\}$ of d orthonormal basis vectors for each cluster j. Select initial values for the disc radii r_j.
(2) Repeat until convergence: {

Date: December 14, 2012.

(a) For all i, j, let $\overline{x}_j^{(i)} = x^{(i)} - \mu_j$. Compute the distance A_{ij} of $\overline{x}_j^{(i)}$ to the subspace spanned by $\{u_{jl}\}$ (equal to the distance of $x^{(i)}$ to the affine space generated by the j-th cluster disc).
We see that $A_{ij} = ||\overline{x}_j^{(i)} - P_j(\overline{x}_j^{(i)})||$, where $P_j(\overline{x}_j^{(i)}) = \sum_{l=1}^d \left\langle \overline{x}_j^{(i)}, u_{jl} \right\rangle$.

(b) Next, for all i, j, compute the distance B_{ij} of $P_j(\overline{x}_j^{(i)})$ to the origin: $B_{ij} = ||P_j(\overline{x}_j^{(i)})||$. We see that if $B_{ij} \leq r_j$, then the distance $D(i, j)$ between $x^{(i)}$ and the j-th disc is A_{ij}.
If $B_{ij} > r_j$, then $D(i, j) = ||\overline{x}_j^{(i)} - \frac{r_j}{B_{ij}} P_j(\overline{x}_j^{(i)})||$.

(c) For all i, set $c^{(i)} := \arg\min_j D(i, j)$.

(d) For all j, set

$$\mu_j := \frac{\sum_{i=1}^m 1\{c^{(i)} = j\} x^{(i)}}{\sum_{i=1}^m 1\{c^{(i)} = j\}}.$$

(e) For all j, set $\{u_{jl}\}$ to be the set of d principal components of $\{x^{(i)} - \mu_j : c^{(i)} = j\}$. (This step involves PCA, i.e., finding the d principal eigenvectors of each cluster's data covariance matrix).

(f) For all j, set $r_j := \max_{i:c^{(i)}=j} ||P_j(x^{(i)} - \mu_j)||$, noting that the meaning of P_j has changed due to the update to $\{u_{jl}\}$.

}

4. Convergence

We can guarantee that k-discs converges to a local optimum. Repeated random initializations are recommended to approximate the global optimum. In specific, we define our objective function $J(c^{(i)}, \mu_j, u_{jl}, r_j) = \sum_{i=1}^m Dist^2(x^{(i)}, \Delta_{c_i})$, where Δ_j is the j-th disc, which can be viewed as a function on μ_j, u_{jl}, r_j. Using the notation from above, $J = \sum_{i=1}^m D(i, c^{(i)})^2$. The value of this function is non-increasing over all iterations.

To see this, we can view the k-discs algorithm as coordinate descent on the arguments of J. (Note that the arguments of J are, more precisely, $\{c^{(i)}\}$, $\{\mu_j\}$, $\{u_{jl}\}$, $\{r_j\}$, though we will omit the curly braces for visual clarity.) Suppose that at some iteration our parameters have values $c^{(i)}$, μ_j, u_{jl}, r_j. We will write their respective updated values as $c'^{(i)}$, μ'_j, u'_{jl}, r'_j. Note that in step (c) we set $c'^{(i)} := \arg\min_j D(i, j)$. Thus $D(i, c'^{(i)})^2 \leq D(i, c^{(i)})^2$ for all i, and $J(c'^{(i)}, \mu_j, u_{jl}, r_j) \leq J(c^{(i)}, \mu_j, u_{jl}, r_j)$.

It is well known that PCA (taking an expanded view of PCA that includes the computation of means μ_j) chooses a set of affine subspaces (or rather, values for μ_j and u_{jl}) which minimize $H(c^{(i)}, \mu_j, u_{jl}) = \sum_{i=1}^m Dist^2(x^{(i)}, S_{c_i})$, where S_j is the affine subspace containing the j-th disc Δ_j. (S'_j and Δ'_j will denote the updated affine subspace and disc.) So $H(c'^{(i)}, \mu'_j, u'_{jl}) \leq H(c'^{(i)}, \mu_j, u_{jl})$. And of course $H(c'^{(i)}, \mu_j, u_{jl}) \leq J(c'^{(i)}, \mu_j, u_{jl}, r_j)$ because $\Delta_j \subset S_j$ for all j. Finally, observe that by the update in step (f), $||P_{c^{(i)}}(x^{(i)} - \mu_{c^{(i)}})|| \leq r'_{c^{(i)}}$ for all i. Hence $Dist^2(x^{(i)}, S'_{c_i}) = Dist^2(x^{(i)}, \Delta'_{c_i})$, and

$$J(c'^{(i)}, \mu'_j, u'_{jl}, r'_j) = H(c'^{(i)}, \mu_j, u_{jl}) \leq J(c^{(i)}, \mu_j, u_{jl}, r_j),$$

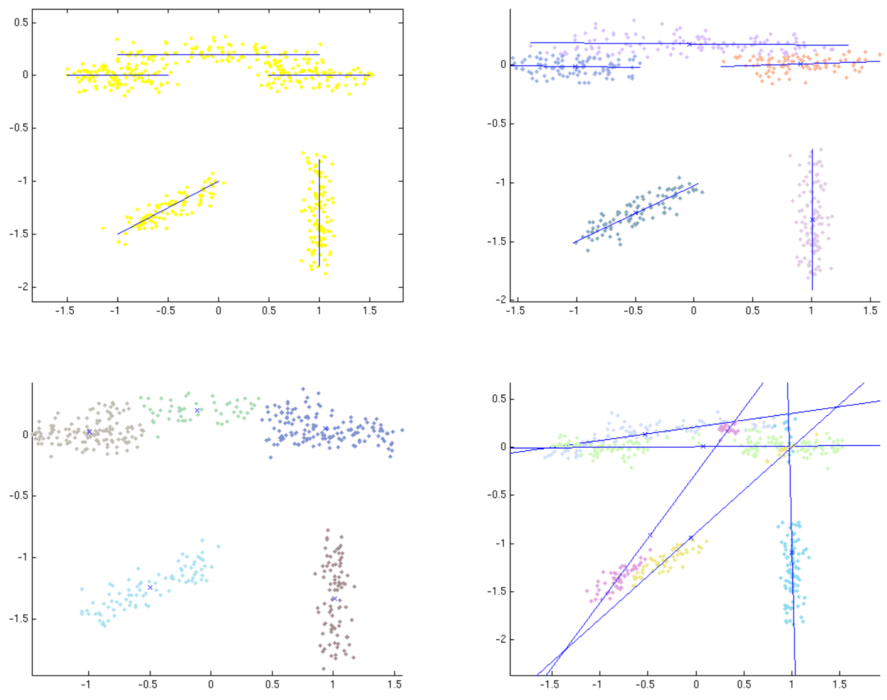
as was to be shown. The objective function is non-increasing and bounded below by zero, hence convergent.

Note that we can slightly modify step (f) of the algorithm and still guarantee convergence of J. For example, we might set $r_j = constant$ for the first several iterations of the algorithm, then update r_j as above. In fact, we can set $r_j = 0$ initially, which is equivalent to running k-means for several iterations. This is a good heuristic for initializing the cluster centroids before running k-discs proper and allows us to skip the PCA step until it is particularly useful. For high-dimensional data, PCA involves finding a (partial) eigendecomposition of an $n \times n$ matrix, which is very expensive. Note also that k-subspaces can be recovered from k-discs by permanently setting $r_j = \infty$.

Allowed to run as initially described, k-discs almost always converged in many fewer than 20 iterations for all data sets tested. However, the convergence behavior depends greatly on the shape of the data as well as initialization procedures. A useful initialization heuristic (to be used before any other processing) which can enhance convergence rates and values is to select special landmark points as clusters. In particular, the maxmin landmark selection procedure simply adds a random point to the landmark set L, then repeatedly adds the point with maximal distance from L to the landmark set L. This procedure gives initial cluster centers that are roughly evenly spaced. This initialization, however, may sometimes bias the clustering algorithm toward a particular set of local minima that are not particularly desirable, and in such cases, repeated random initialization is preferable.

5. Illustrative Synthetic Linear Examples

The following visualizations on the plane demonstrate the improved capacity of k-discs to recover the set of line segments from which a random data set is drawn.

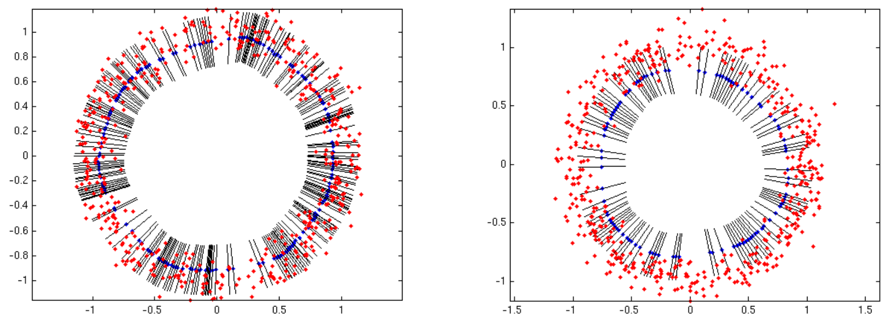


Above left: 500 points sampled from indicated lines with Gaussian noise, $\Sigma = 0.005I$. Above right: Result of k-discs on random sampling. Below left: Result of k-means. Below right: Result of k-subspaces. All clustering initialized with landmark selection.

6. Application to Submanifold Reconstruction

Given a noisy sampling from a d-dimensional submanifold, k-discs (using d-dimensional discs) can be used to generate a piecewise linear approximation of the manifold. The highest value of k that can be used effectively relates inversely with the noisiness of the data. For noisy data, a value of k that is too large will yield cluster centroids that deviate greatly from the submanifold. For $(n-1)$-dimensional submanifolds, if we repeat the k-means algorithm many times and associate each data point with the mean of its associated cluster centroids as well as the mean of its associated normal vectors, we can construct images as below.

This capacity could be of use in computer vision applications in which noisy point cloud representations of surfaces are obtained. The reconstruction points could be used for surface reconstruction as well as topological calculations (of Betti numbers, for example, which could be used for hole detection).



Left: Shown in red are 500 points sampled from the unit circle with Gaussian noise, $\Sigma = 0.01I$. In blue are reconstructed points obtained from averaging 50 repetitions of k-discs with $k = 7$, and in black are the reconstructed normal vectors passing through these points. Right: Same as left, except $\Sigma = 0.05I$, $k = 4$. Landmark initialization was used for both trials. Similar results were obtained for surfaces in three dimensions, such as the sphere and torus, though these cannot be visualized here. Note that for smaller values of k, the radius of the reconstructed circle is significantly smaller than 1. This occurs because the cluster centroids are closer to the origin. Analagous results will occur for any surface of significant curvature. The topology of the surface, however, will be preserved.

7. Kernelization

We observe that k-discs has a natural kernelization. Let ϕ be a mapping from $\mathbb{R}^n$ to some high-dimensional feature space, and suppose that we are given a function $K : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ such that $K(x, y) = \langle \phi(x), \phi(y) \rangle$. We do not work directly with vectors in the high-dimensional space; rather, we can only process them implicitly by using inner products. In particular, the algorithm requires the following inner products to be stored in place of the vectors $\phi(x^{(i)})$, μ_j, u_{jl}:

- $\left\langle \phi(x^{(i)}), \phi(x^{(i')}) \right\rangle = K(x^{(i)}, x^{(i')})$ for all $i, i' = 1, \ldots, m$. These values are stored in a kernel matrix immediately and never updated.
- $\left\langle \phi(x^{(i)}), \mu_j \right\rangle$ for all i, j. These values are updated every time the assignments $c^{(i)}$ are modified. They can be computed

$$\left\langle \phi(x^{(i')}), \mu_j \right\rangle = \left\langle \phi(x^{(i')}), \frac{\sum_{i=1}^m 1\{c^{(i)} = j\}\phi(x^{(i)})}{\sum_{i=1}^m 1\{c^{(i)} = j\}} \right\rangle = \frac{\sum_{i=1}^m 1\{c^{(i)} = j\} K(x^{(i)}, x^{(i')})}{\sum_{i=1}^m 1\{c^{(i)} = j\}}$$

- $\langle \mu_j, \mu_j \rangle$ for all j, updated when the assignments are modified.

$$\langle \mu_j, \mu_j \rangle = \frac{\sum_{i,i'=1}^m 1\{c^{(i)} = j\} 1\{c^{(i')} = j\} K(x^{(i)}, x^{(i')})}{\left(\sum_{i=1}^m 1\{c^{(i)} = j\}\right)^2}$$

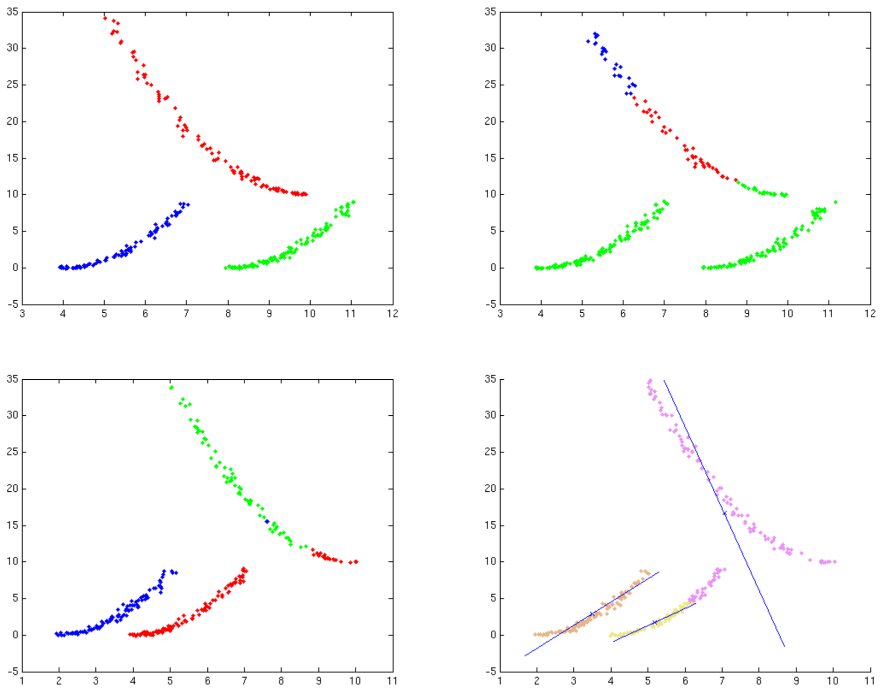
- $\left\langle \phi(x^{(i)}) - \mu_j, u_{jl} \right\rangle$ for all i, j, l. This update is more sophistocated and makes use of the kernel PCA algorithm. Let X_{jl} be the matrix whose rows are $\phi(x^{(i)}) - \mu_{c^{(i)}}$ for all i such that $c^{(i)} = j$. Relabel the rows $\phi(x^{(i)}) - \mu_{c^{(i)}}$ as $\overline{x}^{(p)}$ for $p = 1, \ldots, P$. Regular PCA considers the eigendecomposition of $X_j^T X_j$, but we cannot directly compute this matrix. It can be shown that $X_j X_j^T$ has the same eigenvalues (with all additional eigenvalues equal to zero) and that the (orthonormal) eigenvectors v_{jl} of $X_j X_j^T$ relate to the corresponding eigenvectors u_{jl} of X_j by $u_{jl} = \lambda_{jl}^{-\frac{1}{2}} X^T v_{jl}$. So $\left\langle \phi(x^{(i)}) - \mu_j, u_{jl} \right\rangle = \lambda_{c^{(i)}l}^{-\frac{1}{2}} \langle y, v_{jl} \rangle$, where the p-th component of y is $y_p = \left\langle \phi(x^{(i)}) - \mu_j, \overline{x}^{(p)} \right\rangle$, which can in turn be written in terms of inner products computed above.

The computations in the description of the linear algorithm above can all be carried out using these inner products, but the details are unenlightening and will not be reproduced here. The steps of the kernel k-discs algorithm, including the calculation of r_j, follow from those of the linear algorithm.

We note that the initialization of kernel k-discs is somewhat modified. First, cluster centroids are randomly selected from the original data, then mapped into the high-dimensional feature space by ϕ. We store $\left\langle \phi(x^{(i)}), \mu_j \right\rangle$ and $\langle \mu_j, \mu_j \rangle$ as above. Then clustering assignments are made as in kernel k-means (i.e., using distances in the high-dimensional space from the cluster centroids). Then kernel PCA is performed on each cluster as above to compute $\left\langle \phi(x^{(i)}) - \mu_j, u_{jl} \right\rangle$ for all i, l. The disc radii are then updated in the usual way.

8. Illustrative Synthetic Nonlinear Examples

The following visualizations demonstrate the advantage of k-discs in a particular situation.



All images show 300 points sampled from three quadratic curves with Gaussian noise, $\Sigma = 0.01I$. Above left: Result of kernel k-discs clustering on random sampling, polynomial kernel of degree 2. Above right: Result of kernel k-means, same kernel. Below left: Result of kernel k-subspaces, same kernel. Below right: Typical result of linear k-discs. It is of interest that even though the data is separable with respect to the linear k-discs algorithm, a perfect classification is almost never achieved. Random clustering was used for these trials. (Landmark selection guaranteed poor results in this case.)

9. Conclusions and Further Work

The k-discs algorithm and its kernelization have shown promise in their intended goal of clustering data sampled from bounded submanifolds. Of course, one disadvantage (shared with k-means) of the algorithm is that k must be selected as a parameter. Furthermore, d must also be selected as a parameter, and all discs are constrained to have the same dimension. A more sophistocated version of the algorithm might automatically select dimensions for its cluster descriptions and allow for discs of different dimensions.

Finally, it remains to be seen how k-discs will perform on real-world data. An application of the algorithm to three-dimensional point cloud data could demonstrate the feasibility of using the algorithm for surface reconstruction. It would be interesting to see the results of topological algorithms such as JavaPlex[3] on such reconstructions. In addition, linear and kernel k-discs could be applied to arbitrary clustering problems and compared in performance with k-means, k-subspaces, and any of a multitude of clustering algorithms. Clustering problems are diverse, and the applicability of k-discs in any particular scenario depends mainly on whether the desired clustering has the structure of an affine subspace, either in the original feature space itself or in a high-dimensional feature space that respects a kernel. For example, the success of Li and Fukui[6] in performing facial recognition with linear and kernel k-subspaces suggests that k-discs could have similar, if not better, results for the same type of data.

Assuming that a data set does have this structure, k-discs may sometimes fail converge to or even approximate a global optimum. It would be worthwhile to try to develop some adaptive initialization techniques that improve initialization based on the converged values of the objective function.

References

[1] A. Nielsen, *A Kernel Version of Spatial Factor Analysis*, (2009).
[2] A. Tausz, M. Vejdemo-Johansson, and H. Adams, *JavaPlex: A research software package for persistent (co)homology*, (2011), software available at http://code.google.com/javaplex.
[3] D. Wang, C. Ding, and T. Li, *K-Subspace Clustering*, (2009).
[4] G. Carlsson, *Topology and Data* (2008).
[5] H. Adams and A. Tausz, *Javaplex tutorial*.
[6] X. Li and K. Fukui *Nonlinear k-subspaces based appearances clustering of objects under varying illumination conditions*, (2007).