A New Approach to Data Driven Clustering

Arik Azran ARIK@GATSBY.UCL.AC.UK
Gatsby Computational Neuroscience Unit, University College London, WC1N 3AR, UK

Zoubin Ghahramani ZOUBIN@ENG.CAM.AC.UK
Department of Engineering, University of Cambridge, Cambridge CB2 1PZ, UK

Abstract

We consider the problem of clustering in its most basic form where only a local metric on the data space is given. No parametric statistical model is assumed, and the number of clusters is learned from the data. We introduce, analyze and demonstrate a novel approach to clustering where data points are viewed as nodes of a graph, and pairwise similarities are used to derive a transition probability matrix P for a Markov random walk between them. The algorithm automatically reveals structure at increasing scales by varying the number of steps taken by this random walk. Points are represented as rows of P^t, which are the t-step distributions of the walk starting at that point; these distributions are then clustered using a KL-minimizing iterative algorithm. Both the number of clusters, and the number of steps that 'best reveal' it, are found by optimizing spectral properties of P.

1. Introduction

Clustering is an unsupervised learning problem, where one needs to find *group structure* in a given set of items $S = \{s_n\}_{n=1}^N$. The general approach is to assume that the data set can be organized in K groups, or explained by K factors, and clustering algorithms are designed to find these groups under various possible assumptions. In hard clustering, for example, each point is assigned with a single cluster to form a partitioning $\{S_k\}_{k=1}^K$ such that $\cup_{k=1}^K S_k = S$ and $S_k \cap S_l = \emptyset$ if $k \neq l$. As different problems require different notions of structure it is hard, if not impossible, to have a universal mathematical definition of the goal of clustering that will apply to any possible application. As a result, a pool of methods has been developed, and each has its pros and cons depending on the nature of the data set (see e.g. Jain, Murphy & Flynn, 1999).

In this paper we consider the task of designing an algorithm to cluster a given data set S without any additional information such as the number of clusters K, bounds on the number of points in the k'th cluster $|S_k|$, a parametric statistical model according to which the data is generated etc. Since often there is more than one plausible way to partition the data, the algorithm is required to suggest different 'good' partitionings and associate each of them with a numerical measure that indicates how good it is. Our work was originally inspired by recent contributions in spectral clustering (Meila & Shi, 2001), where data points were viewed as nodes of a fully connected graph, and distance between points was used to define a probability for the transition of a random walk between them. However, whereas spectral methods (Shi & Malik, 2000; Meila & Shi, 2001; Ng, Jordan & Weiss 2001; Kannan, Vempala & Vetta, 2001) rely on the leading eigenvectors, our approach is based on associating each point with a particle which moves between points according to the aforementioned transition matrix, and clustering points is acheived by clustering the distributions of these N particles. This is also different from previous work relating random walk and clustering (e.g. Tishby & Slonim, 2000; Spielman & Teng, 2004), and it provides a new approach to data driven clustering which is both conceptually and practically different from existing methods.

Following a short description of the notation and some known results in Section 2, we define in Section 3 the goal of clustering as it is considered in this paper. In section 4 we introduce and analyze an algorithm for clustering discrete distributions. The behavior of the transition matrix for increasing number of steps is dis-

Appearing in *Proceedings of the 23rd International Conference on Machine Learning*, Pittsburgh, PA, 2006.

cussed in Section 5, leading to an algorithm that finds informative number of steps (Section 6) and the associated number of clusters (Section 7) in the data. The algorithms are demonstrated numerically and some practical issues are discussed in Section 7. The paper is concluded in Section 8.

2. Preliminaries

Assume every member of S is a point in some metric space $\mathcal{X}$ with metric $d(x, y)$. Let $\mathcal{I}_k$ be the indices of all the points in the k'th cluster and $\mathcal{I} = \{\mathcal{I}_k\}_{k=1}^K$ denote the partition. Similarity between pairs of points in S can be measured by any non-negative monotonically decreasing function $w_{mn} = w\left(d(s_m, s_n); \sigma\right)$, with σ being the length scale of w. The smaller it is, the smaller is the 'neighborhood' of a point. While the results in this paper can be applied to any metric space with any function w, for brevity we only discuss $\left(\mathbb{R}^l, \|x-y\|^2\right)$ with the popular Gaussian function

$$w_{mn} = \exp\left(-\frac{\|s_m - s_n\|^2}{\sigma^2}\right) . \tag{1}$$

The pairwise similarities are conveniently summarized in the $N \times N$ *kernel matrix* W, with elements $[W]_{mn} = w_{mn}$ for all $m, n = 1, 2, \ldots, N$. The probability of a particle x to move from point m to point n is given by $P_{mn} = \frac{w_{mn}}{\sum_{n=1}^N w_{mn}}$. This has the intuitive property that as $\|s_m - s_n\|$ increases P_{mn} decreases, and if two points $s_n, s_{n'}$ are in the same distance from s_m then $P_{mn} = P_{mn'}$. Applying this normalization to the entire data set we get the transition matrix P (Meila & Shi, 2001)

$$P = D^{-1}W , \tag{2}$$

where $D = \mathrm{diag}(D_{11}, \ldots, D_{NN})$ is a normalization matrix and $D_{nn} = \sum_{i=1}^N w_{ni}$ is the volume of the n'th node. Notice that each row of P sums to 1, and every entry P_{mn} is nonnegative, satisfying the conditions for the rows to be distributions.

Finally, since P plays an important role in our discussion, we summarize some facts about it.

Lemma 1 *Assume W is full rank and P is given by (2). Let λ_n, v_n be the n'th eigenvalue and eigenvector of P, i.e. $Pv_n = \lambda_n v_n$, and assume without loss of generality that $\lambda_n \geq \lambda_{n+1}$ and $\|v_n\| = 1$. Then,*
1. P is full rank,
2. $\lambda_1 = 1$ and $v_1 = [1, 1, \ldots, 1]^\top / \sqrt{N}$,
3. λ_n is real and $|\lambda_n| \leq 1 \ \forall\ n = 2, 3, \ldots, N$.

The proof is trivial; D is diagonal thus P is defined by normalizing the rows of W and (1) follows, (2) can be verified by direct calculation, the first part of (3) is easily proved using the symmetry of W and the second part by using the equality $\lambda_n v_n(m) = \sum_i P_{mi} v_n(i)$ $\forall n, m$, and for each n choosing $m = \mathrm{argmax}_j |v_n(j)|$.

3. New Criterion for Clustering

3.1. Desiderata for Nonparametric Clustering

Our goal in this paper is to come up with a method for clustering which, given S alone, will automatically find different good partitionings of the data. To this end, we begin by explicitly stating what type of partitioning the algorithm is required to find.

1. If data points are close to each other or they have good paths between them, then they should be clustered together. Conversely, points that are far away from each other and don't have paths between them should be clustered separately.

2. The number of clusters K should be learned directly from the data set, without any assumptions such as their shape or their size $|S_k|$. If there is more than one plausible partitioning then they should all be found.

3. The algorithm should assign all partitionings with an informative quality measure that is both theoretically meaningful and practically useful.

4. If additional knowledge (e.g. the number of clusters K) is given with the data, then it should be easily and naturally combined with the algorithm.

3.2. Data Exploration by Random Walks

Our approach is based on exploring S by letting N particles, each of which begins from a different data point, to move between points according to P. This is motivated by the observation that if there are no good paths between clusters then particles tend to stay within a cluster, independent of the cluster's shape. In addition, since we use N particles simultaneously, then by observing the dynamics of all the particles the number of groups can be found alongside with finding the partition itself (see Figure 1). There are two main questions to be asked here (1) what is the 'right' number of steps that the particles should be allowed to take so as to explore the data effectively and (2) how can the particles 'report back' their findings.

We begin by associating each point s_n with a particle x_n for all $n = 1, 2, \ldots, N$, and consider P to be the transition matrix for these particles. The location of the n'th particle after taking t steps is denoted by $x_n(t)$. So, the distribution of $x_n(1)$ is given by $P_{nm} =$

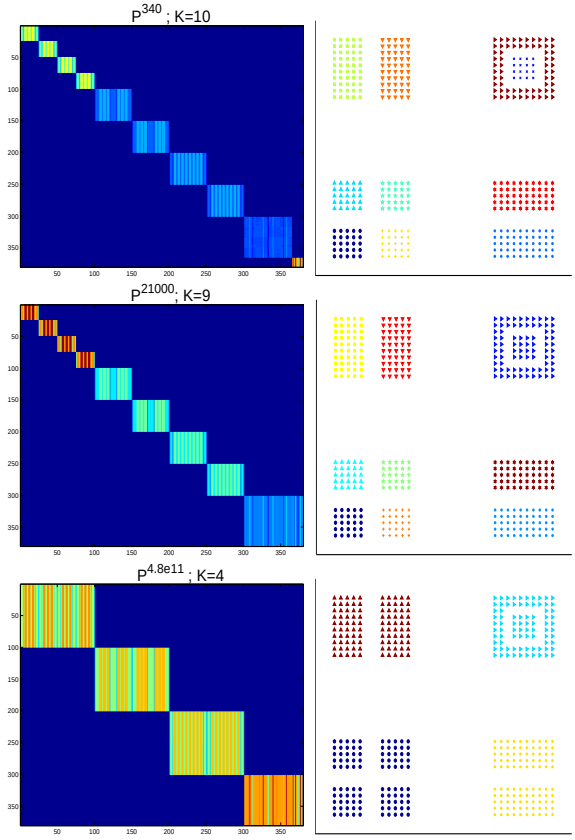

Figure 1. **Demostration of multiscale clustering using markov random walks.** Data set S, comprised of $N = 380$ points, is clustered by clustering the rows of P^t for different values of t. Left column: P^t for $t = 340, 2.1 \cdot 10^5, 4.8 \cdot 10^{11}$, associated with $K = 10, 9, 4$ respectively. Right column: the resulting partitionings of the data (shown as colors). Notice how as t increases clusters in larger scale are revealed and how the distributions of particles, associated with points from the same cluster, are practically the same.

$\mathbb{P}(x_n(1) = s_m)$. More generally, as it is well known from Markov chain theory, the distribution of $x_n(t)$ is given by

$$P^t_{nm} = \mathbb{P}(x_n(t) = s_m) \ . \tag{3}$$

To ease the notation in the sequel we define $\Omega = P^t$ and $\Omega_n = P^t_n$. A key idea in our discussion is the observation that for separated clusters of *any* shape there exist some values of t for which Ω_n is similar to Ω_m if s_n and s_m are in the same cluster. Formally, we consider a parametric mapping $\Psi_t : \mathbb{R}^l \mapsto \mathcal{P}$ of each point $s_n \in \mathbb{R}^l$ to a discrete distribution $\Omega_n \in \mathcal{P} = \{a \in \mathbb{R}^N : a_n \geq 0, \sum_{n=1}^N a_n = 1\}$. If the clusters are well separated, then there exist t such that Ψ_t has the nice property of mapping all of the points in a cluster to almost the same point in the distribution space $\mathcal{P}$, and for K clusters we get K points in $\mathcal{P}$. So, instead of clustering the data points directly, a good partitioning can be found by clustering the distributions for the location of the particles with which they are associated.

4. The K-prototypes Algorithm

In Section 3 the problem of clustering points in $\mathbb{R}^l$ was substituted with clustering the set $\{\Omega_n\}_{n=1}^N$. Although in principle regular K-means can be applied to this problem, it fails to exploit the information conveyed in Ω_n, being a distribution. Euclidean distance is not natural for distributions as it gives all of the elements the same weight. The KL-divergence, on the other hand, gives more weight to elements of the distribution with high probability, which is more suitable for our purposes since usually high probability indicates neighboring points. So, we develop an algorithm that is similar in nature to the K-means algorithm, with the KL replacing the norm-2 metric.

4.1. Algorithm

The goal is to find a set of K distributions $\{Q_k\}_{k=1}^K$, which we refer to as prototypes, and a partitioning $\mathcal{I}$ such that the objective function

$$\begin{aligned} J(Q, \mathcal{I}) &= \sum_{k=1}^{K} \sum_{m \in \mathcal{I}_k} \mathrm{KL}(\Omega_m || Q_k) \\ &= \sum_{k=1}^{K} \sum_{m \in \mathcal{I}_k} \sum_{n=1}^{N} \Omega_{mn} \ln \frac{\Omega_{mn}}{Q_{kn}} \ , \end{aligned} \tag{4}$$

where Q is a $K \times N$ matrix with Q_k as its kth row, is minimized. The direction of the KL in (4) was preferred over the alternative $\mathrm{KL}(Q_k || \Omega_m)$ to encourage good approximations where the entries of Ω_m are away from zero, i.e. for points that are in the same cluster as s_m, which is what we really care about. In addition, this direction of KL is closely related to maximum likelihood modeling of the rows by assuming a multinomial model with mean given by the prototypes.

The summation over k in (4) results in J being a nonconvex function, so that a gradient search is only guarenteed to converge to a local maximum. However, if we assume that some values of the prototypes $Q^{(old)}$ and the partitioning $\mathcal{I}^{(old)}$ are given, then new values $Q^{(new)}, \mathcal{I}^{(new)}$ for which $J\left(Q^{(old)}, \mathcal{I}^{(old)}\right) \geq J\left(Q^{(new)}, \mathcal{I}^{(new)}\right)$ can be found. This is acheived by first leaving $Q^{(old)}$ unchanged and looking for a partitioning $\mathcal{I}^{(new)} = \mathrm{argmin}_{\mathcal{I}} J(Q^{(old)}, \mathcal{I})$ that yields a lower value of J, and then using this partitioning to find a new set of prototypes $Q^{(new)} = \mathrm{argmin}_Q J\left(Q, \mathcal{I}^{(new)}\right)$ that further decrease J.

Input: Transition matrix Ω, number of cluster K
Output: A partitioning $\mathcal{I}$, a matrix of prototypes Q
Algorithm:
0. Initialize: $Q^{(old)}$
1. **Assign points**: for $k = 1, 2, \ldots, K$ set
$\mathcal{I}_k^{(new)} = \left\{ m : k = \text{argmin}_k \, \text{KL}\left(\Omega_m || Q_k^{(old)}\right)\right\}$
2. **Update prototypes**: for $k = 1, 2, \ldots, K$ set
$Q_k^{(new)} = \frac{1}{|\mathcal{I}_k^{(new)}|} \sum_{m \in \mathcal{I}_k^{(new)}} \Omega_m$
3. Set $Q^{(old)} = Q^{(new)}$. If stop condition is satisfied, stop. Otherwise go to 1.

Algorithm 1: K-prototypes

First, it is easy to show (by negation) that

$$\mathcal{I}_k^{(new)} = \left\{ m : k = \underset{k'}{\text{argmin}} \, \text{KL}\left(\Omega_m || Q_{k'}^{(old)}\right)\right\} \quad , \quad (5)$$

that is assigning each Ω_n with Q_k for which $KL(\Omega_m||Q_k)$ is minimized. Next, since $\mathcal{I}^{(new)}$ is assumed to be known when searching for $Q^{(new)}$ then $\min_Q \sum_{k=1}^K \sum_{m \in \mathcal{I}_k^{(new)}} \text{KL}(\Omega_m||Q_k) = \sum_{k=1}^K \min_{Q_k} \sum_{m \in \mathcal{I}_k^{(new)}} \text{KL}(\Omega_m||Q_k)$. Thus, $Q^{(new)}$ can be found by solving the following set of optimization problems

$$\begin{aligned} Q_k^{(new)} &= \underset{Q_k}{\text{argmin}} \sum_{m \in \mathcal{I}_k^{(new)}} \sum_{n=1}^N \Omega_{mn} \ln \frac{\Omega_{mn}}{Q_{kn}} \\ s.t. \quad & \sum_{n=1}^N Q_{kn} = 1, \quad Q_{kn} \geq 0 \; \forall \; n = 1, 2, \ldots, N \end{aligned} \quad (6)$$

for all $k = 1, 2, \ldots, K$. The solution for this problem, found using Lagrange multipliers, is given by

$$Q_k^{(new)} = \frac{1}{\left|\mathcal{I}_k^{(new)}\right|} \sum_{m \in \mathcal{I}_k^{(new)}} \Omega_m \; . \quad (7)$$

To conclude, given the t'th order transition matrix Ω and the number of clusters K, Algorithm 1 finds a partitioning of the rows of Ω into K clusters by executing a simple iterative procedure, similar to K-means, but one that takes into account the fact that the clustered points are discrete distributions.

4.2. Initialization of Q

$Q^{(old)}$ can be initialized (Algorithm 1, step 0) arbitrarily with some random set of K distributions over N states. However, since Algorithm 1 only converges to a local minimum, these values will critically affect the performance of the algorithm. In practice, it is often advantageous to initialize the prototypes so that they will 'cover' the data and not be too close to each other. So, instead of generating random prototypes we can choose any K rows of Ω to initialize $Q^{(old)}$. This ensures that the prototypes are located in regions of $\mathcal{P}$ populated by rows of Ω. Further improvement can be gained by requiring the prototypes, in addition to being part of Ω, to be also far away from each other. Finally, to avoid the danger of only choosing prototypes that lie on the edge of the populated subspace of $\mathcal{P}$ the first prototype can be set to be the mean of Ω. Then, new prototypes are sequentially set to be the row of Ω_n whose distance from its nearest prototype is maximized. This is implemented in Algorithm 2.

Input: Transition matrix Ω, number of cluster K
Output: Star-shaped initialization of Q
Algorithm:
1. Set $Q_1 = \frac{1}{N} \sum_{n=1}^N \Omega_n$ (or any row of Ω).
2. For $k = 2, \ldots, K$ set $Q_k = \Omega_z$ where
$z = \text{argmax}_n \min_{j=1,2,\ldots,k-1} KL(\Omega_n||Q_j)$

Algorithm 2: Star-shaped initialization of Q

5. On the Principal Components of a Markov Random Walk

Consider the spectral decomposition of the transition matrix $P = V \Lambda V^{-1}$ where V is the matrix whose n'th column is v_n and $\Lambda = \text{diag}\,(\lambda_1, \lambda_2, \ldots, \lambda_N)$. Then, $P^t = V \Lambda^t V^{-1}$ and if $\{\lambda_n, v_n\}$ is the eigensystem of P then $\{\lambda_n^t, v_n\}$ is the eigensystem of P^t. This equivalence between P and P^t reveals an important key idea in our analysis - the eigenvalues of P can be used to indicate the scale and quality of partitioning, by separating between the eigenvalues that survive t steps and those that don't. Lemma 2 helps to formulate this by analysing P^t into a weighted sum of N matrices.

Lemma 2 *Let the assumptions and notation of Lemma 1 hold. Then,*
1. for any $t = 1, 2 \ldots$ the matrix P^t is given by

$$P^t = \sum_{n=1}^N \lambda_n^t \frac{v_n v_n^\top D}{v_n^\top D v_n}. \quad (8)$$

2. the set of matrices $\left\{\frac{v_n v_n^\top D}{v_n^\top D v_n}\right\}_{n=1}^N$ with weights $\{\lambda_n^t\}_{n=1}^N$ form an idempotent-orthogonal basis of P^t

The Lemma is proved in Appendix 1. We define an idempotent-orthogonal basis of a matrix as follow.

Definition 1 (Idempotent-Orthogonal Basis) *Assume A is a square full rank matrix of size N,*

and let $\delta_{nm} = 1$ if $n = m$ and 0 otherwise. The set of square matrices of size N, $\{U_n\}_{n=1}^N$, is said to be an idempotent-orthogonal matrix basis of A if $U_n U_m = \delta_{nm} U_n$ and there is a set of N numbers $\{\mu_n\}_{n=1}^N$ such that $A = \sum_{n=1}^N \mu_n U_n$.

An idempotent matrix satisfy $U_n U_n = U_n$, and the orthogonality is due to the condition $U_n U_m = \mathbf{0}$ for $n \neq m$.

We can now give an intuitive interpretation for the spectral decomposition of the markov chain P, analogous to principal component analysis (PCA). In PCA the eigenvectors are pointing in an orthogonal set of directions with maximum variance, and the eigenvalues indicate the variance in these directions. Here, P^t is analyzed into an idempotent-orthogonal basis with weights λ_n^t. The fact that $|\lambda_n| \leq 1$ motivates interpreting the eigenvalues as indicators to the structure in data revealed by the random walk. If λ_n is close to 1, such that λ_n^t is also close to 1, the matrix $\frac{v_n v_n^\top D}{v_n^\top D v_n}$ 'survives' the random walk and is related to stable groups in the data whereas the λ_n^t that tends towards zero are related to decaying basis matrices.

6. Learning t Given K

So far we have assumed that both K and t are known. In this section the number of clusters is still considered to be given, but a value of t which yields a transition matrix P^t that efficiently reveals K clusters should be learned directly from the data. Equation (8) motivates treating λ_n^t as an idication to how stable the partitioning is. Notice that if the graph that represents the data is not fully connected but instead given by a collection of K smaller disconnected graphs (which can be the case if the clusters are well seperated and we threshold the weights), then $\lambda_k = 1$ for all $k = 1, 2, \ldots, K$. In this case, it can be shown that the j'th entry of the (normalizaed) k'th eigenvector is given by $v_k(j) = \frac{1}{\sqrt{|\mathcal{I}_k|}}$ if s_j belong to the k'th cluster and 0 otherwise, for all $k = 1, 2, \ldots, K$ and $j = 1, 2, \ldots, N$. This implies that $\lim_{t\to\infty} P^t = \sum_{k=1}^K \frac{v_k v_k^\top D}{v_k^\top D v_k}$. Motivated from this property, we say that the number of steps t_K effectively reveals K clusters in S if it satisfies

$$t_K = \operatorname*{argmax}_t \left(\lambda_K^t - \lambda_{K+1}^t\right). \tag{9}$$

Solving (9) in general is hard, as each eigenvalue can be positive or negative. However, we can have a closed form solution for a slightly modified version of (9).

First, note that t_K is an estimation for the number of steps that the N particles need to take to discover K clusters in data, and replacing t_K with $t_K \pm 1$ is not expected to make any difference in practice. Thus, we can limit ourselves to even values of t so that (9) can be replaced by

$$t_K = \operatorname*{argmax}_{t \text{ even}} \Delta_K(t)\ , \tag{10}$$

where

$$\Delta_K(t) = |\lambda_K|^t - |\lambda_{K+1}|^t\ , \tag{11}$$

which makes the sign of the eigenvalues irrelevant. Next, (10) can be solved by allowing t to be continuous, and using t_K to be the even integer that is nearest to the solution of

$$\frac{\partial}{\partial t}\Delta_K(t) = 0\ . \tag{12}$$

Recall from Lemma 1 that $\lambda_1 = 1$ and $|\lambda_n| < 1$ for all $n = 2, 3, \ldots, N$ (assuming the graph is fully connected). So, $|\lambda_1|^t - |\lambda_2|^t$ is monotonically increasing and upper bounded by 1, and $\Delta_K(t)$ is unimodal for all $K = 2, 3, \ldots, N-1$. Lemma 3 can be shown from (12) and gives an analytical solution for (10).

Lemma 3 *The solution for (10) is given by*

$$t_K = \left[\frac{\ln\left(\frac{\ln(|\lambda_{K+1}|)}{\ln(|\lambda_K|)}\right)}{\frac{\ln(|\lambda_K|)}{\ln(|\lambda_{K+1}|)}}\right]_{\text{even}} \tag{13}$$

for all $K = 2, 3, \ldots, N-1$, where $[\mathrm{z}]_{\text{even}}$ is the nearest even integer to z, and $t_1 \to \infty$.

7. Learning K and t From the Data

7.1. Algorithm

The question that is asked in this section is *how can good pairs of (K, t_K) be found from the data*, rather than what is a good t_K for a given K. The idea is as follows; different plausible K can be revealed by increasing t, and the plausibility of the revealed number of clusters K can be measured by $\Delta_K(t)$. To be more percise, if some K is indeed a plausible value, then there should be some t_K such that the eigengap has a local maxima at t_K.

To formalize it, define the maximal eigengap

$$\Delta(t) = \max_k \Delta_k(t) \tag{14}$$

for every even t, and the number of clusters that is best revealed, as a function of t, by the location of the maximal eigengap

$$\mathcal{K}(t) = \operatorname*{argmax}_k \Delta_k(t)\ . \tag{15}$$

Input: Data set S
Output: M partitioning of S with associated stability and plausibility measures $\left\{\{S_k^m\}_{k=1}^{K_m}, \alpha_m\right\}_{m=1}^{M}$.
Algorithm:
1. Compute P according to (2), and find its spectrum.
2. Find $\mathcal{T}$ and denote $M = |\mathcal{T}|$.
3. For every $m = 1, 2 \ldots, M$ call Algorithm 1 with $\Omega = P^{\mathcal{T}_m}$ and $K = \mathcal{K}(\mathcal{T}_m)$ to find a partitioning $\{S_k^m\}_{k=1}^{K_m}$.
4. Grade partitioning plausibility by $\alpha_m = \Delta(\mathcal{T}_m)$.

Algorithm 3: Multiscale K-prototype clustering

Note the close relation between (15) and (10), where the former can be viewed as turning the latter on its head. Notice that generally (i) the scale of data exploration increases with t, and as a result the revealed number of clusters $\mathcal{K}(t)$ is expected to decrease, and (ii) there is usually more than a single value of t for which (15) yields the same answer. Next, to choose the number of steps t_K, define the set of t over which the same number of clusters K is revealed

$$T_K = \{t : \mathcal{K}(t) = K\} \ . \tag{16}$$

Among all members of T_K, the number of steps that best reveals K clusters is $t_K = \text{argmax}_{t \in T_K} \Delta_k(t)$.

The results of applying this to a data set S, comprised of $N = 71$ images of hand written digits, are given in Figure 2. Each image has 8×8 pixels, thus each data point is a member of $\mathbb{R}^{64}$. Notice how $\Delta(t)$ and $\mathcal{K}(t)$, shown in the bottom row, couple the search for t with the search for K such that $\Delta(t)$ has local maxima if the partition into K clusters is plausible. In particular, $\Delta(t)$ has 3 local maxima at $t_{4,3,2} = 60, 9 \cdot 10^3, 2.4 \cdot 10^7$, and $\mathcal{K}(t_{4,3,2}) = 4, 3, 2$ respectively. Notice how $\Delta(t_{4,3,2}) = 0.8, 1, 1$ can be used to indicate the plausibility of the resulting partitionings and how the number of steps $|T_{4,3,2}|$, associated with each partitioning, can be used to indicate how stable it is. Notice also how the matrices $P^{t_{4,3,2}}$ give an intuitive two dimensional description of how each image relates to the rest of the data set after exploring it with $t_{4,3,2}$ steps, which is ideal for the K-prototypes algorithm.

Figure 3 shows the performance of Algorithm 3 applied to a data set which consists of 300 noisy and randomly rotated digits with 3 labels (100 images per label). Each image is given by 26×26 pixels and is represented as a point in $\mathbb{R}^{676}$. Due to the random rotation of the digits, where the rotation is uniformly distributed over $[0^o, 360^o)$, this data set is formed of rings in $\mathbb{R}^{676}$. Applying K-means to this data set severely mixes labels 2 and 5, indicating that the associated

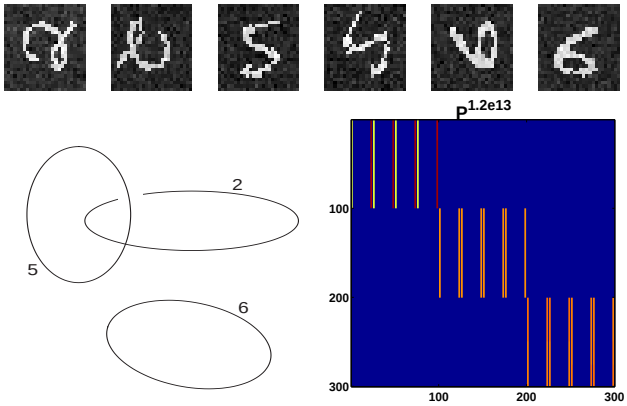

Figure 3. **Digits data set.** Algorithm 3 applied to a set of 300 randomly rotated digits with 3 labels. The nature of this data set is of interlocked rings (see text). The algorithm automatically finds the number of clusters and a perfect partitioning of the data.

rings are interlocked.

7.2. Analytical Computation of t_K

In step 2 of Algorithm 3, the set

$$\mathcal{T} \equiv \{t : \Delta(t) \text{ is a local maxima}\} \tag{17}$$

of all local maxima of $\Delta(t)$ needs to be found. This can be done easily by using the fact that $\mathcal{T}$ is a subset of $\{t_K\}_{K=2}^{N-1}$, each element of which is found[1] analytically according to Lemma 3. There is no need to scan over all values of t.

7.3. Hierarchical Algorithm and Comparison With Other Methods

Algorithm 3 can be easily adjusted to find a single hierarchical partitioning by modifying steps 3,4. First, in step 3 only the smallest number of plausible clusters (smallest member of $\mathcal{T}$) is used to call algorithm 1. Then, Algorithm 3 is called recursively for each of the resulting clusters. Applying this idea to the digits data set, for example, yields a tree which first splits S into $S_1 = A \cup B \cup C$ and $S_2 = D$, then S_1 is split into $S_{11} = A \cup B$ and $S_{12} = C$ etc (see Figure 2). The results of running our algorithm on another data set, comprised of 100 face images[2], are given in Figure 4.

It is interesting to test other algorithms on these data sets. However, since our algorithm finds both the hierarchy and the cluster assignment of the data whereas other algorithms usually assume some side information, a full comparison is hard to obtain. So, we used

[1] In practice it is sufficient to use some $K_{max} < N$.
[2] available at http://www.cs.toronto.edu/roweis/data.html

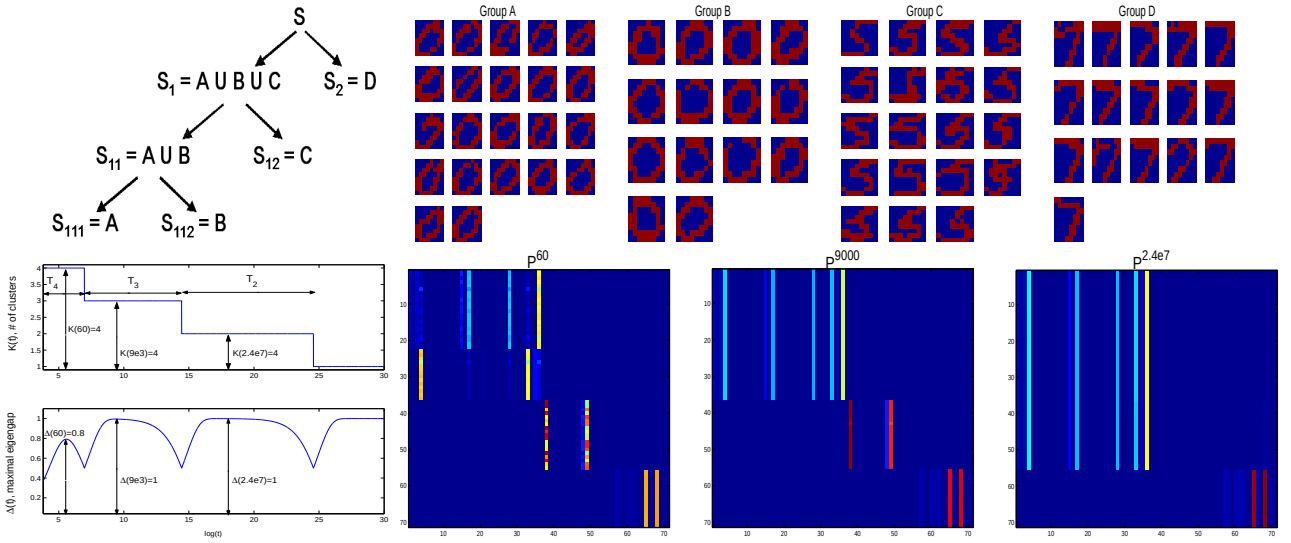

Figure 2. Results of applying Algorithm 4 to a data set $S = A \cup B \cup C \cup D$ of $N = 71$ images of digits shown in the bottom, each of which given by 8×8 pixels. Notice how the local maxima of $\Delta(t)$ and the assoiated $K(t)$ indicate plausible number of clusters, and how P^{t_K} gives an intuitive description of the revealed partitioning. For $K = 2$ we get a partitioning $\{A \cup B \cup C, D\}$, for $K = 3$ the partitioning is $\{A \cup B, C, D\}$ and for $K = 4$ it is $\{A, B, C, D\}$. The top-left figure shows the results of the hierarchical algorithm. Notice how different levels of the tree correspond to different scales of partitioning. For more details consult the text.

the tree structure found by our algorithm and used other algorithms to find the cluster assignment for this structure. We applied the algorithms in (Meila & Shi, 2001) and in (Ng et al, 2001) to all the data sets used here. For the former the results were identical to ours and the latter produced partitionings that were slightly less consistent with the known labels. This gives an empirical indication for the advantage of our algorithm, where in addition to producing comparable cluster assignments, it also learns the tree structure from the data.

7.4. Some Implementation Issues

Initializing the lengthscale σ. The role of σ is to reliably capture the local structure in the data, so that by increasing t the global structure can be revealed. Hence choosing a good value for this parameter is crucial for the success of the algorithm. Although we could scan over σ to find a value that optimizes some desired property (e.g. maximal eigengap), in practice we found that setting σ to be smaller than 99% of $\{\|s_m - s_n\|, m \neq n\}_{m,n=1}^{N}$ leads to good results.

Similarity of a point and itself. According to (1) $w_{nn} = 1$ and $P_{nn} > P_{nm}$ for all $m \neq n$. This is desirable if we wish outliers to be made into singleton clusters. Otherwise, setting $w_{nn} = 0$ forces the particle to leave the point and explore its neighborhood. In our experiments we chose the later setting.

Sparsifying W. Small σ results in most of W's entries being nearly zero, and in practice sparsifying W (e.g. by thresholding its entries or by only keeping a fixed small number of neighbors for each point) is not expected to significantly change the results. For sparse W, the complexity of computing the spectral decomposition of P can be significantly eased, for example using Lanczos method (for efficient implementation in MATLAB® see also the command eigs).

Complexity analysis. In terms of execution time, computing the KL divergence costs $O(N)$ for distributions of length N. Thus, algorithms 2 cost $O(KN^2)$ and algorithms 1 cost $O(cKN^2)$, where c is the number of iterations in algorithm 1 (typically $c \ll N$) . Since P is sparse, then step 1 of Algorithm 3 is $O(N^2)$, and thus Algorithm 3 is $O(cMKN^2)$. As for the memory requirements, all algorithms require $O(N^2)$ values to be stored. These costs are comparable to other affinity propagation methods such as spectral clustering.

8. Conclusion

We have introduced and analyzed a novel algorithm for nonparametric clustering where data points are associated with particles and multiscale clustering is achieved by clustering the distributions for the loca-

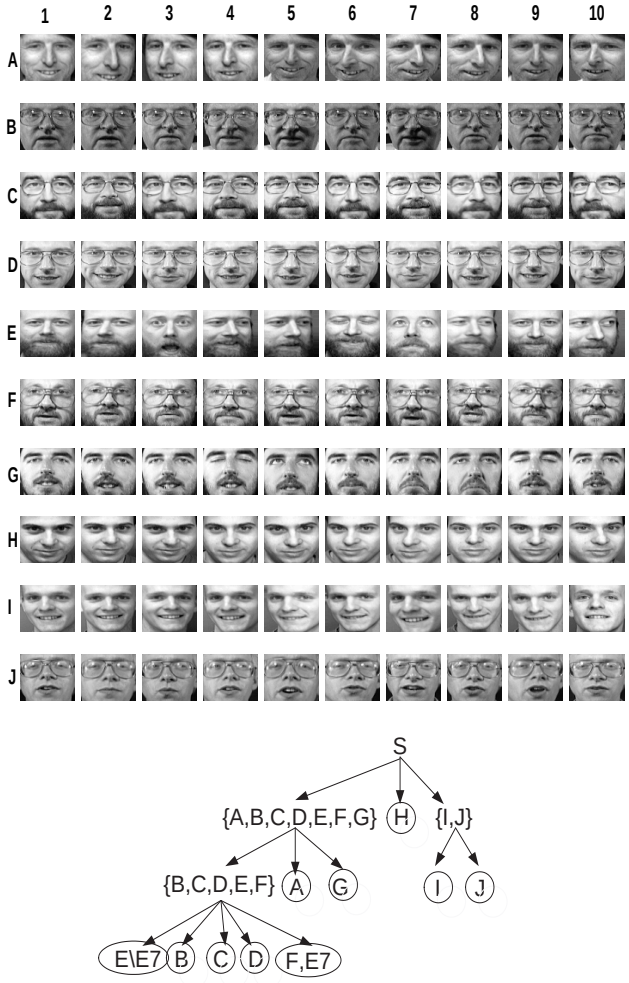

Figure 4. Results of applying the hirarchical algorithm to a set of 100 face images. The number of clusters was inferred from the data and all images, except E7, were clustered according to their subjects.

tion of these particles. If the number of clusters K is known, then Algorithm 1 can be used to find a partitioning of the data. If it is not given with the data, then Algorithm 4 finds several plausible values for this parameter, calls Algorithm 1 to find a partitioning for each of these values and grade each of the resulting partitionings. This is all achieved given the data and a metric on the feature space alone. We wish to emphasize that although the algorithm is motivated by searching exaustively over the number of steps t, the analysis yields a closed form solution for all the parameters, so the algorithm is especially easy to implement. In addition, a new algorithm for clustering distributions over a finite number of states was derived and analyzed, and links to Markov chain random walks, the well known PCA decomposition and spectral graph properties were established.

A. Proof of Lemma 2

We give a short version of the proof, as the long one is somewhat tedious and left for the full paper. We wish to express $P = V\Lambda V^{-1}$ in terms of V alone (no dependence on V^{-1}). First, let us rewrite $D^{-1}Wv = \lambda v$ as $D^{-\frac{1}{2}}WD^{-\frac{1}{2}}D^{\frac{1}{2}}v = \lambda D^{\frac{1}{2}}v$. Since $D^{-\frac{1}{2}}WD^{-\frac{1}{2}}$ is symmetric then it has orthogonal eigenvectors

$$\left(D^{\frac{1}{2}}v_n\right)^\top D^{\frac{1}{2}}v_m = v_n^\top Dv_m = \delta_{mn}v_n^\top Dv_n, \qquad (18)$$

which also implies $\left[V^\top DV\right]_{mn} = \delta_{mn}v_n^\top Dv_n$. Next, notice that we can write

$$\begin{aligned} P &= V\Lambda V^{-1} = V\Lambda V^{-1}\left(V^\top D\right)^{-1}V^\top D \\ &= V\Lambda\left(V^\top DV\right)^{-1}V^\top D\ . \end{aligned} \qquad (19)$$

Since both $V^\top DV$ and Λ are diagonal then $\Lambda\left(V^\top DV\right)^{-1}$ is also diagonal with elements $\frac{\lambda_n}{v_n^\top Dv_n}$ and some simple matrix multiplication complete the proof of the first part of the lemma.

The second part can be proved by verification. The result of multiplying two basis matrices is given by $\frac{v_nv_n^\top D}{v_n^\top Dv_n}\frac{v_mv_m^\top D}{v_m^\top Dv_m} = \frac{v_nv_n^\top Dv_mv_m^\top D}{(v_n^\top Dv_n)(v_m^\top Dv_m)} = \frac{v_nv_n^\top D}{v_n^\top Dv_n}\delta_{mn}$, where the second equality is due to (18). □

References

Jain, A. K., Murty, M. N., & Flynn, P. J. (1999). Data clustering: a review. *ACM Computing Surveys*, *31*, 264–323.

Kannan, R., Vempala, S., & Vetta, A. (2000). On Clustering: Good, Bad and Spectral. *Proceedings of the 41st Annual Symposium on the Foundation of Computer Science.*

Meila, M., & Shi, J. (2001). A random walks view of spectral segmentation. *Tenth International Workshop on Artificial Intelligence and Statistics (AISTATS).*

Ng, A. J., Jordan, M. I., & Weiss, Y. (2001). On spectral clustering: Analysis and an algorithm. *Advances in Neural Information Processing Systems (NIPS),14.*

Shi, J., & Malik, J. (2000). Normalized Cuts and Image Segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence.*

Spielman, D., & Teng, S. (2004). Nearly linear time algorithms for graph partitioning, graph sparsification, and solving linear systems. *36th Annual ACM Symposium on Theory of Computing* (pp. 81–90).

Tishby, N., & Slonim, N. (2000). Data clustering by markovian relaxation and the information bottleneck method. *NIPS* (pp. 640–646).