Multi-way Clustering Using Super-Symmetric Non-negative Tensor Factorization

Amnon Shashua, Ron Zass, and Tamir Hazan

School of Engineering and Computer Science, The Hebrew University, Jerusalem
{shashua, zass, tamirr}@cs.huji.ac.il

Abstract. We consider the problem of clustering data into $k \geq 2$ clusters given complex relations — going beyond pairwise — between the data points. The complex n-wise relations are modeled by an n-way array where each entry corresponds to an affinity measure over an n-tuple of data points. We show that a probabilistic assignment of data points to clusters is equivalent, under mild conditional independence assumptions, to a super-symmetric non-negative factorization of the closest hyper-stochastic version of the input n-way affinity array. We derive an algorithm for finding a local minimum solution to the factorization problem whose computational complexity is proportional to the number of n-tuple samples drawn from the data. We apply the algorithm to a number of visual interpretation problems including 3D multi-body segmentation and illumination-based clustering of human faces.

1 Introduction

We address the fundamental problem of grouping feature vectors (points) on the basis of multi-wise similarity or coherency relationships among n-tuples of points. The case of pairwise ($n = 2$) relationships has drawn much attention in statistical, graph theoretical and computer vision literature. For example, a clustering task of a collection of points $\mathbf{x}_1, ..., \mathbf{x}_m$ in Euclidean space R^n may be induced by a symmetric "affinity" matrix $K_{ij} = e^{-\|\mathbf{x}_i - \mathbf{x}_j\|^2/\sigma^2}$ which would serve as the input to a process aimed at assigning the m points into $k \geq 2$ classes. The greatly popular "spectral" clustering technique, for example, looks for the k leading eigenvectors of a normalized version of K as a new coordinate system which in ideal settings would map the original coordinates of the points to k points in R^k, one per each cluster [10, 11]. Graph theoretical methods perform normalization on the affinity matrix (producing the Laplacian of K) whereby the second smallest eigenvector splits the points into two clusters [15, 8], and more recently it was shown that conditionally independent statements on the unknown labels given the data points lead to the finding that $K = GG^\top$, $G \geq 0$, where G contains the probabilistic assignments of points to clusters [20].

It has been recently pointed out by a number of authors [1, 5, 21] that for many computer vision and machine learning applications a pairwise affinity relationship among points does not capture the complexity of the problem. For example, if a parametric model requires d points for a definition, then $n \geq d+1$

points can be used to provide an affinity value by taking the square residual error Δ^2 of the least-squares fit of the n points to the model and translating it into a probability value $\kappa(\mathbf{x}_{i_1}, ..., \mathbf{x}_{i_n}) = e^{-\Delta^2/\sigma^2}$, where $1 \leq i_1, ..., i_n \leq m$. The affinities form an n-way (tensor) super-symmetric array $K_{i_1,...,i_n} = \kappa(\mathbf{x}_{i_1}, ..., \mathbf{x}_{i_n})$ which like as above is the input for a clustering of the m points into $k \geq 2$ clusters. Computer vision applications for parametric models include (i) 3D-from-2D multi-body segmentation where under an affine model one would need $n \geq 5$ points to determine an affinity value [17] and under a perspective model $n \geq 9$ points are required [9]; (ii) segmenting 3D objects taken from the same pose but under varying illumination conditions — for matte surfaces ignoring self-shadowing one would need $n \geq 4$ pictures for determining an affinity, i.e., the likelihood that the four pictures are of the same 3D surface [13], and (iii) multi-model selection in general.

We address in this paper the problem of clustering m points into $k \geq 2$ clusters given an n-way super-symmetric affinity array $K \in [m] \times .. \times [m] = [m]^{\times n}$. We will first describe the state of the art in this domain and then proceed to describe our contribution.

1.1 Previous Work on n-Way Clustering and Our Approach

Clustering from an n-way affinity array is new to computer vision and machine learning but has been a topic of extensive research in VLSI and PCB clustering placement since the early 70s. A convenient representation of the problem is given by a *hypergraph*, with m vertices and $\binom{m}{n}$ different hyper-edges, where the vertices correspond to the points (circuit elements in VLSI) to be clustered into $k \geq 2$ parts and the hyper-edges (nets connecting circuit elements) correspond to subsets of vertices where the degree n of an edge is the number of vertices spanned by it.

The techniques employed by the VLSI/PCB community for hypergraph partitioning into clusters are largely heuristic in nature — for a review see [2]. The recent work coming out from the vision and machine learning communities [1, 5, 21] seek an approximate graph that best resembles the original hypergraph. For example, [1] define a pairwise affinity as a weighted average over all n-tuple affinities containing the two points in question — this can be viewed as a projection of the original tensor K onto a two-dimensional matrix by a weighted sum of its the slices. Similarly, [5] defines a pairwise affinity between points $\mathbf{x}_r, \mathbf{x}_s$ as a sum of products $K_{r,i_2,...,i_n} K_{s,i_2,...,i_n}$ over all $i_2, ..., i_n$. Finally, [21] performs a multiplicative normalization with the vertices degrees (the sum of weights incident to a vertex) as part of creating a Laplacian of the hypergraph. Both [1, 21] are consistent with graph theoretical research which define hypergraph Laplacians by summing up all the weights incident to pairs of vertices [12], while the work of [5] is inspired by "high order SVD" literature (referenced therein).

The idea of projecting the hypergraph onto a graph is not without merit. However, the projection from a high-order affinity array to a pairwise affinity would have a high SNR for *simple* problems. Problems with a small number of clusters having a high number of points per cluster relative to the affinity degree would benefit from the projection approach. Generally, however, a projection induces information-loss and the pairwise affinities will get increasingly obscured with increasing affinity degree — and since we have here a “curse of dimensionality” effect, a rapid decline of pairwise affinity SNR is expected with increasing problem complexity.

Rather than performing a projection we work with the full affinity tensor. Our approach enables us to define any affinity degree we desire — including one obtained by projection of the original tensor to a lower degree one, and in particular to a pairwise affinity. Starting from a super-symmetric tensor K of any degree, we show that a general probabilistic argument on conditional independence introduces a simple connection between K and the desired $m \times k$ probabilistic partition matrix $G \geq 0$. The connection is two fold (i) the “balancing” requirement on the cluster sizes requires K to be hyper-stochastic, and (ii) G is obtained by a super-symmetric non-negative factorization (SNTF) of K. The algorithm we derive for performing the SNTF is based on a positive-preserving gradient descent scheme. The scheme also supports partial sampling of the affinity tensor which is necessary since it is practically impossible to fill in, or even store, a full high-degree affinity array. The complexity of the update step is $O(mkp)$ where $p \leq \binom{m}{n}$ is the number of samples.

The work presented here is an outgrowth of our algebraic treatment of pairwise affinity clustering showing that K is completely positive [20] and of a general treatment of tensor ranks and conditional independence with latent variables [14].

2 Probabilistic Clustering from n-Way Affinity Arrays

Let $\mathcal{D} = \{\mathbf{x}_1, ..., \mathbf{x}_m\}$ be points in R^d which we wish to assign to k clusters $C_1, .., C_k$ and let $y_i \in \{1, ..., k\}$ be the associated (unknown) labels. We assume that we have a way to measure the probability, which for now is simply an affinity measure in the range $(0, 1]$, that any n-tuple of points $\mathbf{x}_{i_1}, ..., \mathbf{x}_{i_n}$, $1 \leq i_j \leq m$, belong to the same cluster. For example, if we know that the clusters are defined as $n-1$ dimensional subspaces, then $k(\mathbf{x}_{i_1}, ..., \mathbf{x}_{i_n}) = e^{-\Delta}$, where Δ is the volume defined by the n-tuple, would be a reasonable measure of n-tuple affinity.

Given the affinities $k(\mathbf{x}_{i_1}, ..., \mathbf{x}_{i_n})$, which form an n-way array K indexed by $K_{i_1,...,i_n}$, we wish to assign a probability $g_{r,s} = P(y_s = r \mid \mathcal{D})$ of point $\mathbf{x}_s$ belonging to cluster C_r. The desired membership probabilities form a non-negative $m \times k$ matrix $G = [\mathbf{g}_1, ..., \mathbf{g}_k]$, thus our goal is to find G given K. We will derive below an algebraic constraint on the n-way array K and relate it, by means of factorization and linear constraints, to the desired matrix G.

Consider the labels y_i as *latent* variables and assume that $y_1 \perp ... \perp y_m \mid \mathcal{D}$, i.e., that the labels are independent of each other given the entire set of data points. Then, the probability $P(y_{i_1} = r, ..., y_{i_n} = r \mid \mathcal{D})$ that $\mathbf{x}_{i_1}, ..., \mathbf{x}_{i_n}$ belong to cluster C_r, is factorizable:

$$P(y_{i_1} = r, ..., y_{i_n} = r \mid \mathcal{D}) = P(y_{i_1} = r \mid \mathcal{D}) \cdots P(y_{i_n} = r \mid \mathcal{D}).$$

The probability that the n-tuple are clustered together is given by marginalization:

$$K_{i_1,...,i_n} = \sum_{r=1}^{k} P(y_{i_1} = r \mid \mathcal{D}) \cdots P(y_{i_n} = r \mid \mathcal{D}) = \sum_{r=1}^{k} g_{r,i_1} \cdots g_{r,i_n},$$

which translate to the fact that K should be a rank=k super-symmetric tensor:

$$K = \sum_{r=1}^{k} \mathbf{g}_r^{\otimes n}, \qquad \mathbf{g}_r \geq 0,$$

where $\mathbf{g}^{\otimes n}$ denotes the rank-1 tensor $\mathbf{g} \otimes \mathbf{g} \otimes ... \otimes \mathbf{g}$. In other words, the cluster assignment probabilities are related to a non-negative super-symmetric factorization of the input n-way array K. To complete the algebraic relation between K and G we need to consider the constraints on K such that the n-way affinity array will indeed represent a distribution:

Proposition 1. *Given uniform priors on the distribution of labels, i.e., $P(y_i = j) = 1/k$ for all $i = 1, ..., m$, the n-way array K must be hyper-stochastic:*

$$\sum_{i_1,..,i_{j-1},i_{j+1},...,i_n} K_{i_1,...,i_n} = \left(\frac{m}{k}\right)^{n-1} \mathbf{1}, \qquad j = 1, ..., n$$

where $\mathbf{1}$ *is the m-dimensional vector* $(1, ..., 1)$.

Proof: From the definition of G we have that the rows sum to 1: $\sum_r P(y_s = r \mid \mathcal{D}) = \sum_r g_{rs} = 1$. Therefore, the uniform priors means that each column sums to m/k: $\sum_s g_{rs} = m/k$. The rows and columns sums propagate to a (scaled) hyper-stochastic constraint on K:

$$\begin{aligned}
\sum_{i_1,..,i_{j-1},i_{j+1},...,i_n} K_{i_1,...,i_n} &= \sum_{r=1}^{k} g_{r,i_j} \sum_{i_1,..,i_{j-1},i_{j+1},...,i_n} g_{r,i_1} \cdots g_{r,i_{j-1}} g_{r,i_{j+1}} \cdots g_{r,i_n} \\
&= \sum_{r=1}^{k} g_{r,i_j} \left(\sum_{i_1} g_{r,i_1}\right) \cdots \left(\sum_{i_{j-1}} g_{r,i_{j-1}}\right) \left(\sum_{i_{j+1}} g_{r,i_{j+1}}\right) \cdots \left(\sum_{i_n} g_{r,i_n}\right) \\
&= \left(\frac{m}{k}\right)^{n-1} \sum_{r=1}^{k} g_{r,i_j} = \left(\frac{m}{k}\right)^{n-1}
\end{aligned}$$

$\square$

Note that the hyper-stochasticity constraint is "balanced partitions" in disguise. The uniform prior assumption in fact constraints the dataset to form k "balanced" clusters. Since we do not wish to *enforce* strictly a balanced partition we will seek only a "soft" version of the hyper-stochastic constraint by adopting the following scheme: (i) find a hyper-stochastic approximation F to the input affinity array K, and (ii) given F, perform a super-symmetric non-negative tensor factorization (SNTF), i.e., find $\mathbf{g}_1, ..., \mathbf{g}_k \geq 0$ that minimize the Frobenius norm $\|F - \sum_r \mathbf{g}_r^{\otimes n}\|^2$.

Finding a hyper-stochastic approximation to K can be done by repeating a normalization step which is an extension of the symmetrized Sinkhorn [16, 20] rows and columns normalization procedure for matrices. The following proposition forms a normalization algorithm which converges to a super-symmetric hyper-stochastic array:

Proposition 2. *For any non-negative super-symmetric n-way array $K^{(0)}$, iterating the process:*

$$K^{(t+1)}_{i_1,...,i_n} = \frac{K^{(t)}_{i_1,...,i_n}}{(a_{i_1}\cdots a_{i_n})^{1/n}},$$

where

$$a_i = \sum_{i_2,...,i_n} K_{i,i_2,...,i_n}, \qquad i = 1, ..., m$$

converges to a hyper-stochastic array.

The proof is in Appendix B. In the pairwise affinity ($n = 2$) case, the results above state that $K = GG^\top$ and that prior to factorizing K we should normalize it by replacing it with $F = D^{-1/2}KD^{-1/2}$ where D is a diagonal matrix holding the row sums of K. If we iterate this normalization procedure we will obtain a doubly-stochastic approximation to K. This is consistent with [20] which argues that the conditional independence statements $y_i \perp y_j \mid \mathcal{D}$ lead to the finding that $K = GG^\top$ which also underlies the k-means, spectral clustering and normalized cuts approaches. In other words, the conditional independence assumptions we made at the start are already built-in into the conventional pairwise affinity treatment — we have simply acknowledged them and extended them beyond pairwise affinities.

3 The SNTF Algorithm

We are given a n-way affinity array $K \in [d_1] \times .... \times [d_n]$ with $d_i = m$ being the number of data points to be clustered. An entry $K_{i_1,...,i_n}$ with $1 \leq i_j \leq m$ denotes the (un-normalized) probability of the n-point tuple $\mathbf{x}_{i_1}, ..., \mathbf{x}_{i_n}$ to be clustered together. The tensor K is super-symmetric because the probability $K_{i_1,...,i_n}$ does not depend on the order of the n points. Furthermore, we can ignore entries with repeating indices and focus only on the case $i_1 \neq ... \neq i_n$ (this is crucial for the success of the algorithm). For practical reasons, we would like to store only a single representative of each n-tuple (instead of $n!$ entries), thus we focus only on the entries $i_1 < i_2 < ... < i_n$. Accordingly, we define the order-restricted Frobenius (semi) norm:

$$\|K\|^2_o =< K, K >_o = \sum_{1 \leq i_1 < i_2 < ... < i_n \leq m} K^2_{i_1,...,i_n},$$

where $< A, B >_o$ is the inner-product (restricted to strictly ascending order) operation. Note that when K is super-symmetric then

$$\|K\|^2_o = \frac{1}{n!} \sum_{i_1 \neq ... \neq i_n} K^2_{i_1,...,i_n}$$

which is the restriction of the Frobenius norm to non-repeating indices. As mentioned in the previous section, we pass K through a normalization process and obtain a normalized version denoted by F. Our goal is to find a non-negative matrix $G_{m\times k}$ whose columns are denoted by $\mathbf{g}_1, ..., \mathbf{g}_k$ such as to minimize the following function:

$$f(G) = \frac{1}{2}\|F - \sum_{j=1}^{k} \mathbf{g}_j^{\otimes n}\|_o^2,$$

We derive below a positive-preserving update rule: $g_{r,s} \leftarrow g_{r,s} - \delta_{rs}\partial f/\partial g_{r,s}$. We start with the derivation of the partial derivative $\partial f/\partial g_{r,s}$. The differential df is derived below:

$$df = d\frac{1}{2} < F - \sum_{j=1}^{k}\mathbf{g}_j^{\otimes n} \ , \ F - \sum_{j=1}^{k}\mathbf{g}_j^{\otimes n} >_o = < \sum_{j=1}^{k}\mathbf{g}_j^{\otimes n} - F \ , \ d(\sum_{j=1}^{k}\mathbf{g}_j^{\otimes n}) >_o$$
$$= < \sum_{j=1}^{k}\mathbf{g}_j^{\otimes n} - F \ , \ \sum_j (d\mathbf{g}_j) \otimes \mathbf{g}_j^{\otimes(n-1)} + \mathbf{g}_j \otimes (d\mathbf{g}_j) \otimes \mathbf{g}_j^{\otimes(n-2)} + ... + \mathbf{g}_j^{\otimes(n-1)} \otimes d\mathbf{g}_j >_o$$

The partial derivative with respect to $g_{r,s}$ (the s'th entry of $\mathbf{g}_r$) is:

$$\frac{\partial f}{\partial g_{rs}} = < \sum_{j=1}^{k}\mathbf{g}_j^{\otimes n} - F \ , \ \mathbf{e}_s \otimes \mathbf{g}_r^{\otimes(n-1)} + ... + \mathbf{g}_r^{\otimes(n-1)} \otimes \mathbf{e}_s >_o$$

where $\mathbf{e}_s$ is the standard vector $(0, 0, .., 1, 0, ..0)$ with 1 in the s'th coordinate. It will be helpful to introduce the following notation: let $1 \le i_2 < ... < i_n \le m$ and let $1 \le s \le m$ be different from $i_2, ..., i_n$, then $s \rightarrow i_2, .., i_n$ is an ascending n-tuple index (i.e., s is inserted into $i_2, ..., i_n$ in the appropriate position). Thus, for example:

$$< F, \mathbf{a}\otimes\mathbf{b}\otimes\mathbf{b} + \mathbf{b}\otimes\mathbf{a}\otimes\mathbf{b} + \mathbf{b}\otimes\mathbf{b}\otimes\mathbf{a} >_o = \sum_{i_1 \neq i_2 < i_3} F_{i_1 \rightarrow i_2, i_3} a_{i_1} b_{i_2} b_{i_3}$$

Using the above short-hand notation, the partial derivative becomes:

$$\frac{\partial f}{\partial g_{r,s}} = \sum_{j=1}^{k} g_{j,s} \sum_{s \neq i_2 < ... < i_n} \prod_{q=2}^{n} g_{j,i_q} g_{r,i_q} - \sum_{s \neq i_2 < ... < i_n} F_{s \rightarrow i_2, .., i_n} \prod_{q=2}^{n} g_{r,i_q} \quad (1)$$

We will be using a "positive preserving" gradient descent scheme $g_{rs} \leftarrow g_{rs} - \delta_{rs}\partial f/\partial g_{rs}$. Following [7] we set the gradient step size δ_{rs} as follows:

$$\delta_{rs} = \frac{g_{rs}}{\sum_{j=1}^{k} g_{j,s} \sum_{s \neq i_2 < ... < i_n} \prod_{q=2}^{n} g_{j,i_q} g_{r,i_q}} \quad (2)$$

After substitution of eqn. 2 into the gradient descent equation we obtain a multiplicative update rule:

$$g_{rs} \leftarrow \frac{g_{rs} \sum_{s \neq i_2 < ... < i_n} F_{s \rightarrow i_2, .., i_n} \prod_{q=2}^{n} g_{r,i_q}}{\sum_{j=1}^{k} g_{j,s} \sum_{s \neq i_2 < ... < i_n} \prod_{q=2}^{n} g_{j,i_q} g_{r,i_q}} \quad (3)$$

The update rule preserves positivity, i.e., if the initial guess for G is non-negative and F is super-symmetric and non-negative, then all future updates will maintain non-negativity. The proof that the update rule reduces $f(G)$ and converges to a local minima is presented in Appendix A.

There are a couple of noteworthy points to make. First, removing from consideration entries in F that correspond to repeated indices makes the energy function $f(g_{r,s})$ be quadratic (when all other entries of G are fixed) which in turn is the key for the update rule above to reduce the energy at each step. Second, each sample of n-tuple corresponds to $n!$ entries of the affinity tensor K. As the dimension grows, any algorithm for processing K becomes unpractical as simply recording the measurements is unwieldy. The scheme we presented above records only the $\binom{m}{n}$ entries $1 \leq i_1 < ... < i_n \leq m$ instead of m^n in return for keeping a lexicographic order during measurement recording and during the update process of $g_{r,s}$ (access to $F_{s \rightarrow i_2,...,i_n}$).

Next, for large arrays, the need to sample all the possible (ordered) n-tuples out of m points introduces an excessive computational burden. In fact, it is sufficient to sample only a relatively small fraction of all n-tuples for most clustering problems. The sampling introduces vanishing entries in K that do not correspond to low affinity of the corresponding n-tuple but to the fact that the particular tuple was not sampled — those should be weighted-out in the criteria function $f(G)$. A "weighted" version of the scheme above requires merely a straightforward modification of the update rule:

$$g_{rs} \leftarrow \frac{g_{rs} \sum_{s \neq i_2 < ... < i_n} W_{s \rightarrow i_2,...,i_n} F_{s \rightarrow i_2,..,i_n} \prod_{q=2}^{n} g_{r,i_q}}{\sum_{j=1}^{k} g_{j,s} \sum_{s \neq i_2 < ... < i_n} W_{s \rightarrow i_2,...,i_n} \prod_{q=2}^{n} g_{j,i_q} g_{r,i_q}} \tag{4}$$

where $W_{i_1,...,i_n} \geq 0$, $i_1 < ... < i_n$, is a weight associated with the entry $K_{i_1,...,i_n}$. In particular we are interested in the binary weighting scenario where the weight is zero if the n-tuple $\mathbf{x}_{i_1}, ..., \mathbf{x}_{i_n}$ was not sampled and '1' otherwise. To summarize, the n-way clustering algorithm is presented below:

1. Construct K: sample n-tuples $\mathbf{x}_{i_1}, ..., \mathbf{x}_{i_n}$, $i_1 < ... < i_n$, and set $K_{i_1,...,i_n} = k(\mathbf{x}_{i_1}, ..., \mathbf{x}_{i_n})$. Set $W_{i_1,...,i_n} = 1$.
2. Normalize K: apply the iterative normalization scheme which generates F (Prop. 2).
3. Factor F: starting with an initial guess for $G \geq 0$, iteratively update the entries $g_{r,s}$ one at a time using eqn. 4 until convergence is reached.

Note that only sampled entries participate in the algorithm, therefore the complexity of each update step (eqn. 4) is a constant factor of the number of samples. The complexity of the algorithm is $O(mkp)$ where $p \leq \binom{m}{n}$ is the number of samples (number of non-vanishing entries of W).

4 Experiments

We begin by studying the performance of the SNTF algorithm on synthetic data compared to the graph projection methods [1, 5, 21]. A comparative study

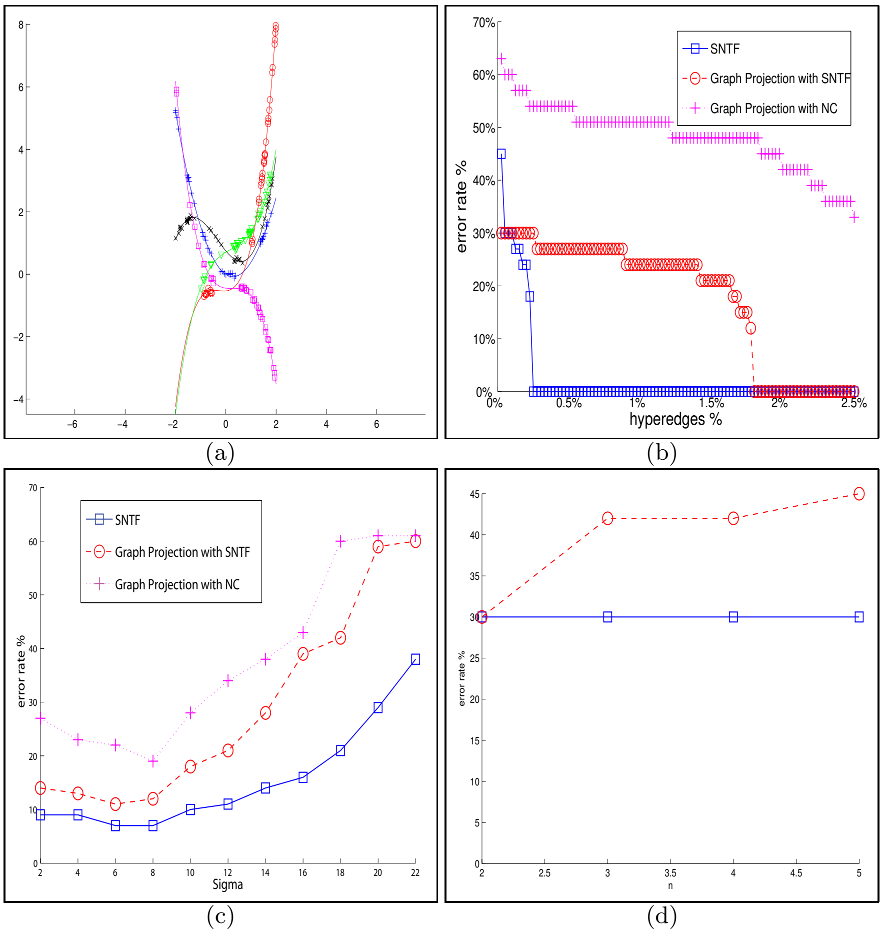

Fig. 1. Synthetic study of clustering $m = 200$ points arranged in $k = 5$ 3rd-order curves (i.e., affinity degree is $n = 5$). See text for details on each display.

of graph projection against outlier rejection algorithms (like RANSAC) and the multi-level hypergraph partitioning algorithms used in the VLSI community was presented in [1] showing a significant advantage to graph projection. Therefore we will focus our comparative study on the performance relationship between SNTF and graph projection.

The graph projection approximates the original hypergraph with a graph followed by spectral clustering. In practice, when the affinity degree n is large one needs to use sampling, i.e., during the projection not all hyper-edges are used since their number grows exponentially with the affinity degree ([5] addressed the sampling issue). We expect the graph projection to work well when the problem is "simple", i.e., when a projection from $\binom{m}{n}$ hyper-edges to $\binom{m}{2}$ edges can be done with minimal information loss – in those cases it is worthwhile to reduce the problem size from a hypergraph to a graph rather than working directly with

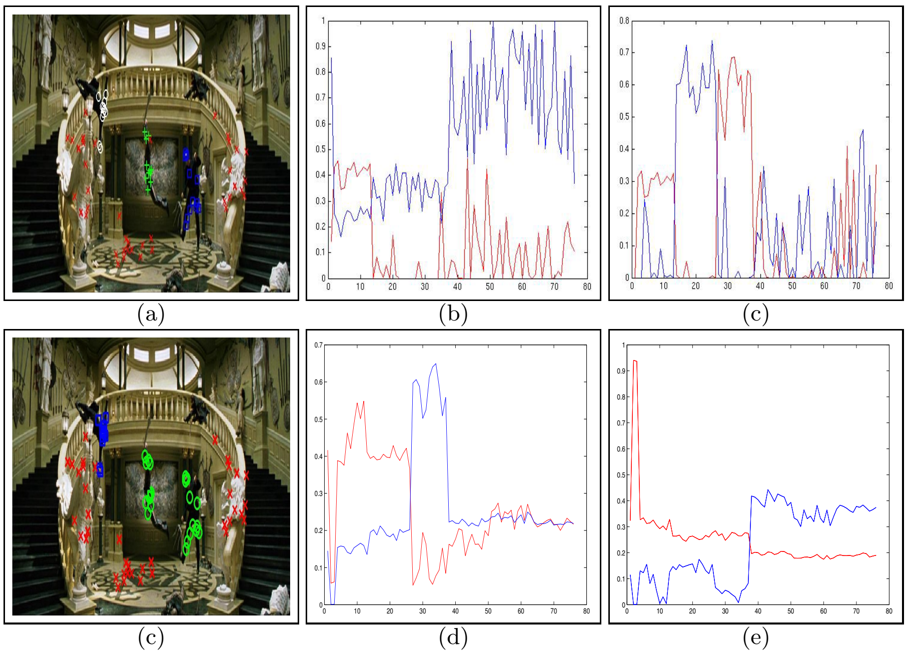

Fig. 2. 3D-from-2D motion segmentation. (a) shows a picture with 76 points over four separate bodies, (b,c) show the resulting four columns of the partition matrix G using SNTF with a 9-way affinity array. The bottom row shows the results after projecting the affinity array onto a matrix. The projection resulted in significant information-loss which caused performance degradation.

the full affinity tensor. On the other hand, when the number of points is large or when the affinity degree is high, one would expect a significant information-loss during projection with a resulting degraded performance.

In our first experiment we generated $m = 200$ points in the 2D plane laying on $k = 5$ 3rd-order polynomials with added Gaussian noise. The number of hyper-edges (entries of the affinity tensor K) is $\binom{200}{5}$ and since a 3rd-order 1D polynomial is determined by four coefficients we have $n = 5$. We ran SNTF, graph projection using Normalized-Cuts (NC) and graph projection using SNTF (i.e., the same algorithm described in this paper but for $n = 2$). We varied the runs according to the sampling percentage ranging from $0.02\% - 2.5\%$ of sampled hyper-edges. Fig. 1a shows the input data and Fig. 1b shows the clustering error percentage of the three runs per sampling. The error of the SNTF is indeed higher than the graph projection when the sampling is very low (0.02%), i.e., when the affinity tensor is very sparse and thus the projection onto a graph (matrix) does not suffer from information-loss. As the sampling rate increases the performance of the SNTF on $n = 5$ original affinity tensor significantly outperforms both graph projection runs and reaches perfect clustering much earlier (0.2% compared to 1.5% sampling). Fig. 1c compares the error rate of

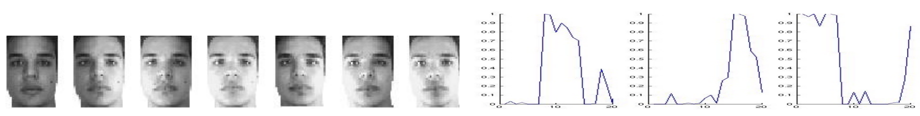

Fig. 3. Segmenting faces under varying illumination conditions. See text.

SNTF and graph projections (NC and SNTF with $n = 2$) using 0.15% sampling rate while varying σ used in computing the affinity from the residual Δ, i.e., $e^{-\Delta^2/\sigma^2}$. One can see that the SNTF on the original affinity degree $n = 5$ consistently outperforms clustering over graph projections — regardless of the clustering technique.

It is possible to use the SNTF framework in coarse-to-fine manner by generating affinity tensors of degree $q = 2, 3, ..., n$ by means of projection. Starting from $q = 2$ (graph) we recover the partition matrix G and use it as the initial guess for the SNTF of level $q + 1$ and so forth. In other words, the SNTF framework allows the flexibility to work with projections of the original affinity tensor, but instead of being limited to a projection onto a graph we could work with any affinity degree. Fig. 1d shows the percentage of error on the same data but with 0.02% sampling (where we have seen that the graph projection has the upper-hand) using the coarse-to-fine approach. One can see that the error remains fixed compared to an increasing error for each projection level when the SNTF does not use the resulting partition matrix of the previous level as an initial guess. This also confirms that there is a tradeoff between the complexity of the energy landscape introduced in high-degree affinities and the information loss introduced by aggressive projections. Ideally, one should work with a projection to the smallest affinity degree with minimal information loss. The advantage of the SNTF framework is that we are free to choose the affinity degree, whereas with graph projection the affinity degree is set to $n = 2$.

We move next to a 3D motion segmentation experiment. Fig. 2a shows a frame from "Matrix Reloaded" where we track 76 points arranged on four different moving bodies: the background (moving due to camera motion) and three separate people moving independently from the background motion. The points were tracked across two successive frames and our task is to perform a segmentation (clustering) of the points and assign each point to the proper moving body. It is well known that under perspective projection each pair of matching points p_i, p'_i in the image plane represented in homogenous coordinates satisfy a bilinear constraint: $p'^{\top}_i F p_i = 0$ where F is a 3×3 matrix iff the corresponding 3D points are part of a single moving object [9]. Therefore, we need $n = 9$ points in order to obtain an affinity measurement, i.e., the likelihood that the 9-tuple arise form the same moving object. The affinity tensor has $\binom{76}{9}$ entries and we sample roughly one million entries from it with a proximity bias, i.e., once a point is sampled the next point is biased towards close points according to a Normal distribution. We ran SNTF with $k = 4$ clusters on the 9-degree (sampled) affinity tensor. Fig. 2b,c shows the four columns of the partition matrix G. Recall that the entries of each column represent the assignment probability of the corresponding point to the cluster associated with the column. The values of G induce a clear-cut segmentation of the points to four separate bodies and the assignments are shown in Fig. 2a as varying color and shape. This particular segmentation problem is sufficiently challenging both for the graph projection approach and to the geometric-specific methods of [19, 18]. With regard to graph projection, the projection from a 9-degree affinity to a pairwise affinity is very aggressive with significant information-loss. Fig. 2e,f shows the four columns of G recovered from SNTF with $n = 2$ (followed by a projection) — one can see that one of the moving bodies got lost.

Finally we ran an experiment on segmenting faces under varying illumination conditions. It is well known that under certain surface property assumptions (Lambertian) the space of pictures of a 3D object ignoring cast-shadows lie in a 3D subspace [13]. We therefore need a 4th-degree affinity measured over quadruples of pictures. Fig. 3 shows a sequence of pictures of a person under varying illumination conditions adopted from the AR dataset. We had 21 pictures spanning three different persons and we ran SNTF using 4-degree affinity tensor with $k = 3$ clusters. The three columns of the partition matrix G are shown in the right display. The pictures are unambiguously assigned to the correct person. Similar results of comparable quality were also obtained by graph projection.

Acknowledgments

We thank Alex Samorodnitsky for assistance in proving Proposition 2 and acknowledge the financial support from ISF grant 189/03 and GIF grant 773/2004.

References

1. S. Agrawal, J. Lim, L. Zelnik-Manor, P. Perona, D. Kriegman, and S. Belongie. Beyond pairwise clustering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2005.
2. C.J. Alpert and A.B. Kahng. Recent directions in netlist partitioning. *The VLSI Journal*, 19(1-2):1–81, 1995.
3. C.M. Fiduccia and R.M. Mattheyses. A linear time heuristic for improving network partitions. In *Proc. of the 19th IEEE Design Automation Conference*, pages 175–181, 1982.
4. I.M. Gelfand, M.M. Karpanov, and A.V. Zelevinsky. *Discriminants, Resultants and multidimensional determinants.* Birkhauser Boston, 1994.
5. V.M. Govindu. A tensor decomposition for geometric grouping and segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2005.
6. B.W. Kernighan and S. Lin. An efficient heuristic procedure for partitioning graphs. *The Bell System Technical Journal*, 49(2), 1970.
7. D. Lee and H. Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401:788–791, 1999.
8. T. Leighton and S. Rao. An approximate max-flow min-cut theorem for uniform multicommodity flow problems with applications to approximate algorithms. In *Proceedings Symposium on Foundations of Comp. Sci.*, 1988.

9. H.C. Longuet-Higgins. A computer algorithm for reconstructing a scene from two projections. *Nature*, 293:133–135, 1981.
10. A.Y. Ng, M.I. Jordan, and Y. Weiss. On spectral clustering: Analysis and an algorithm. In *Proceedings of the conference on Neural Information Processing Systems (NIPS)*, 2001.
11. P. Perona and W. Freeman. A factorization approach to grouping. In *Proceedings of the European Conference on Computer Vision*, 1998.
12. J.A. Rodriguez. On the Laplacian eigenvalues and metric parameters of hypergraphs. *Linear and Multilinear Algebra*, 50(1):1–14, 2002.
13. A. Shashua. Illumination and view position in 3D visual recognition. In *Proceedings of the conference on Neural Information Processing Systems (NIPS)*, Denver, CO, December 1991.
14. A. Shashua and T. Hazan. Non-negative tensor factorization with applications to statistics and computer vision. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2005.
15. J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8), 2000.
16. R. Sinkhorn. A relationship between arbitrary positive matrices and doubly stochastic matrices. *Ann. Math. Statist.*, 35:876–879, 1964.
17. S. Ullman and R. Basri. Recognition by linear combination of models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13:992–1006, 1991.
18. R. Vidal, Y. Ma, S. Soatto, and S. Sastry. Two-view multibody structure from motion. *International Journal of Computer Vision*, 2004.
19. L. Wolf and A. Shashua. Two-body segmentation from two perspective views. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Hawaii, Dec. 2001.
20. R. Zass and A. Shashua. A unifying approach to hard and probabilistic clustering. In *Proceedings of the International Conference on Computer Vision*, Beijing, China, Oct. 2005.
21. D. Zhou, J. Huang, and B. Scholkopf. Beyond pairwise classification and clustering using hypergraphs. Technical report, Max Planck Institute for Biol. Cybernetics, TR-143, August 2005.

A Proof of Convergence: The Update Rule

Let $f(g_{rs})$ be the energy as a function of g_{rs} (all other entries of G remain constant) and let g'_{rs} be the updated value according to eqn. 3. We wish to show that if we make a gradient descent with a step size δ_{rs} given by eqn. 2 (which as we saw leads to a positive-preserving update), then $f(g'_{rs}) \leq f(g_{rs})$. They key is that δ_{rs} is smaller than the inverse second derivative:

Proposition 3. *The update scheme $g'_{rs} = g_{rs} - \delta_{rs}\partial f/\partial g_{rs}$, with δ_{rs} given by eqn. 2 and the partial first derivative is given by eqn. 1, reduces the optimization function, i.e., $f(g'_{rs}) \leq f(g_{rs})$.*

Proof: The second derivative is:

$$\frac{\partial^2 f}{\partial g_{rs} \partial g_{rs}} = \sum_{s \neq i_2 < \ldots < i_n} \prod_{q=2}^{n} g^2_{r,i_q},$$

and the step size δ_{rs} satisfies:

$$\delta_{rs} = \frac{g_{rs}}{\sum_{j=1}^{k} g_{j,s} \sum_{s \neq i_2 < \ldots < i_n} \prod_{q=2}^{n} g_{j,i_q} g_{r,i_q}} \leq \frac{g_{rs}}{g_{r,s} \sum_{s \neq i_2 < \ldots < i_n} \prod_{q=2}^{n} g_{r,i_q}^2} = \frac{1}{\partial^2 f / \partial g_{rs} \partial g_{rs}}$$

The Taylor expansion of $f(g_{rs} + h)$ with $h = -\delta_{rs} \partial f / \partial g_{rs}$ is:

$$f(g'_{rs}) = f(g_{rs}) - \delta_{rs} (\frac{\partial f}{\partial g_{rs}})^2 + \frac{1}{2} \delta_{rs}^2 (\frac{\partial f}{\partial g_{rs}})^2 \frac{\partial^2 f}{\partial g_{rs} \partial g_{rs}},$$

from which follows:

$$f(g_{rs}) - f(g'_{rs}) = \delta_{rs} (\frac{\partial f}{\partial g_{rs}})^2 (1 - \frac{1}{2} \delta_{rs} \frac{\partial^2 f}{\partial g_{rs} \partial g_{rs}}) \geq 0,$$

since $\delta_{rs} \partial^2 f / \partial g_{rs} \partial g_{rs} \leq 1$. $\square$

We apply the update rule in a Gauss-Seidel fashion according to a row-major raster scan of the entries of G (a row-major raster scan has the advantage of enabling efficient caching). Since the energy is lower-bounded, twice differentiable, and is monotonically decreasing via the update rule, yet cannot decrease beyond the lower bound (i.e., positive preserving), then the process will converge onto a local minimum of the optimization function $\frac{1}{2} \| F - \sum_{j=1}^{k} g_j^{\otimes n} \|^2$ with entries with repeated indices ignored.

B Proof of Convergence: Normalization Scheme

We prove the following proposition:

For any non-negative super-symmetric n-way array $K^{(0)}$, without vanishing slices, iterating the process:

$$K^{(t+1)}_{i_1,\ldots,i_n} = \frac{K^{(t)}_{i_1,\ldots,i_n}}{(a_{i_1} \cdots a_{i_n})^{1/n}}, \tag{5}$$

where

$$a_i = \sum_{i_2,\ldots,i_n} K_{i,i_2,\ldots,i_n}, \qquad i = 1, \ldots, m$$

converges to a hyper-stochastic array.

Proof: we define the *hyper-permanent* (following the definition of hyper-determinant [4]):

$$hperm(K) = \sum_{\sigma_2 \in S_m} \cdots \sum_{\sigma_n \in S_m} \prod_{i=1}^{m} K_{i,\sigma_2(i),\ldots,\sigma_n(i)},$$

where S_m is the permutation group of m letters. Let K' be the n-way array following one step of the normalization step described in eqn. 5. We have:

$$\prod_{i=1}^{m}(a_i a_{\sigma_2(i)} \cdots a_{\sigma_n(i)})^{1/n} = \prod_{i=1}^{m}(a_i^n)^{1/n} = \prod_{i=1}^{m} a_i,$$

from which we can conclude that:

$$hperm(K') = \frac{1}{\prod_{i=1}^{m} a_i} hperm(K).$$

To show that the normalization scheme monotonously increases the hyper-permanent of the n-way array we need to show that $\prod_{i=1}^{m} a_i \leq 1$. From the arithmetic-geometric means inequality it is sufficient to show that $\sum_{i=1}^{m} a_i \leq m$. From the definition of a_i we have:

$$\sum_{i=1}^{m} a_i = \sum_{i,i_2,\ldots,i_n} K_{i,i_2,\ldots,i_n} \frac{1}{(a_i a_{i_2} \cdots a_{i_n})^{1/n}}. \tag{6}$$

From the arithmetic-geometric means inequality $(\prod_{i=1}^{m} x_i)^{1/m} \leq (1/m)\sum_i x_i$, replace x_i with $1/a_i$ (recall that $a_i > 0$) and obtain:

$$\frac{1}{(a_1 a_2 \cdots a_m)^{1/m}} \leq \frac{1}{m}\sum_{i=1}^{m}\frac{1}{a_i},$$

and in general for any n-tuple $1 \leq i_1 < ... < i_n \leq m$:

$$\frac{1}{(a_{i_1} \cdots a_{i_n})^{1/n}} \leq \frac{1}{n}(\frac{1}{a_{i_1}} + ... + \frac{1}{a_{i_n}}). \tag{7}$$

By substituting the inequality eqn. 7 into eqn. 6 while noting that:

$$\sum_{i,i_2,..,i_n} K_{i,i_2,\ldots,i_n}\frac{1}{a_{i_j}} = \sum_{i_j=1}^{m}\frac{1}{a_{i_j}} \sum_{i,i_2,\ldots,i_{j-1}i_{j+1},\ldots,i_n} K_{i,i_2,\ldots,i_{j-1}i_{j+1},\ldots,i_n} = m,$$

we obtain that $\sum_i a_i \leq m$ as required. Therefore, we conclude so far that each step of the normalization scheme increases the hyper-determinant of the previous step. The hyper-permanent is bounded from above since:

$$hperm(K) \leq \prod_{i=1}^{m} a_i \leq 1,$$

therefore the process must converge. The process converges when *hperm* $(K') = hperm(K)$ which can happen only of $a_1 = ... = a_m = 1$. $\square$