Kernelizing Sorting, Permutation and Alignment for Minimum Volume PCA

Tony Jebara

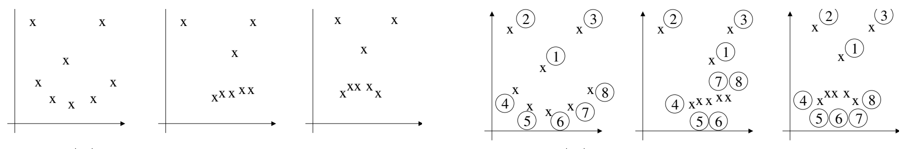
Columbia University, New York, NY 10027, USA
jebara@cs.columbia.edu

Abstract. We propose an algorithm for permuting or sorting multiple sets (or bags) of objects such that they can ultimately be represented efficiently using kernel principal component analysis. This framework generalizes sorting from scalars to arbitrary inputs since all computations involve inner products which can be done in Hilbert space and kernelized. The cost function on the permutations or orderings emerges from a maximum likelihood Gaussian solution which approximately minimizes the volume data occupies in Hilbert space. This ensures that few kernel principal components are necessary to capture the variation of the sets or bags. Both global and almost-global iterative solutions are provided in terms of iterative algorithms by interleaving variational bounding (on quadratic assignment problems) with a Kuhn-Munkres algorithm (for solving linear assignment problems).

1 Introduction

Sorting or ordering a set of objects is a useful task in practical unsupervised learning as well as in general computation. For instance, we may have a set of unordered words describing an individual's characteristics in paragraph form and we may wish to sort them in a consistent manner into fields such that the first field or word describes the individual's eye color, the second word describes his profession, the third word describes his gender, and so forth. Alternatively, as in Figure 1, we may want to sort or order dot-drawings of face images such that the first dot is consistently the tip of the nose, the second dot is the left eye, the third dot is the right eye and so forth. However, finding a meaningful way to sort or order sets of objects is awkward when the objects are not scalars (scalars can always be sorted using, e.g. quick-sort). We instead propose sorting many bags or sets of objects such that the resulting sorted versions of the bags are easily representable using a small number of kernel principal components. In other words, we will find the sorting or ordering of many bags of objects such that the manifold formed by these sorted bags of objects will have low dimensionality.

In this article, we refer to sorting or ordering in the relative sense of the word and seek the relative ordering between objects in two or more unordered sets. This is equivalent to finding the correspondence between multiple sets of objects. A classical incarnation of the correspondence task (also referred to as



(a) 3 Unsorted dot images (b) 3 Sorted dot images

Fig. 1. Sorting or matching of 3 bags of 8 (x, y) coordinates representing faces.

matching, permutation or ordering between sets) is the so-called linear assignment problem (LAP). A familiar example of LAP is in an auction or garage-sale where N goods are available and N consumers each attribute a value to each good. This solution to LAP is the the best pairing of each consumer to a single good such that the total value obtained is maximal. This is solvable using the classical Kuhn-Munkres algorithm in $O(N^3)$ time. Kuhn-Munkres provides a permutation matrix capturing the relative ordering between the two sets (goods and consumers).

Recent efficient variants of Kuhn-Munkres make it practical to apply to bags of thousands of objects [3]. Alternatively, relaxations of LAP have been proposed including the so-called invisible hand algorithm [8]. These tools have been used for finding correspondence and aligning images of, for instance, digits [2, 14] to obtain better models (such as morphable or corresponded models). In fact, handling permutable or unordered sets is relevant for learning and image classification as well. For example, permutable images and other objects have been handled via permutationally invariant kernels for support vector machine classifiers [7] or permutationally invariant expectation-maximization frameworks [6]. It is known that removing invariant aspects of input data (such as permutation) can improve a learning method [13]. Another approach is to explicitly estimate the ordering or permutation by minimizing the number of principal components needed to linearly model the variation of many sets or bags of objects [5, 4].

In this paper, we build up a novel algorithm starting from the Kuhn-Munkres algorithm. Kuhn-Munkres sorts only a pair of bags or sets containing N vector-objects such that we minimize their squared norm. Our novel algorithm upgrades the search for an ordering from two bags to many simultaneous bags of objects by iterating the Kuhn-Munkres algorithm with variational bounds. The iterations either minimize the squared norm from all sorted bags to a common "mean bag" or minimize the dimensionality of the resulting manifold of sorted bags. These two criteria correspond to a generalization of the linear assignment problem and to the quadratic assignment problem, respectively. Both are handled via iterative solutions of the Kuhn-Munkres algorithm (or fast variants). We also kernelize the Kuhn-Munkres algorithm such that non-vectorial objects [11] can also be ordered or sorted.

2 Permuting several sets

Consider a dataset $\mathcal{D}$ of T sets or bags $\mathcal{D} = \{\chi_t\}_{t=1}^T$. Each of these bags is merely a collection of N unordered objects $\chi_t = \{\gamma_{t,n}\}_{n=1}^N$. We wish to find an ordering for objects in these bags that makes sense according to some fairly general criterion. However, in the general case of bags over unusual objects (vectors, strings, graphs, etc.) it is not clear that a natural notion of ordering exists a priori. We will exploit kernels since they have been shown to handle a diverse range of input spaces. If our sorting algorithms leverage these by exclusively using generalized inner products within sorting computations we would be able to sort a variety of non-scalar objects. We therefore propose another criterion for sorting that finds orderings. The criterion is that the resulting ordered bags can be *efficiently* encoded using principal components analysis (PCA) or kernel principal component analysis (kPCA) [12]. Essentially, we want kPCA to capture the variation seen in the dataset with as few dimensions as possible.

We will eventually deal with non-vectorial objects but for simplicity, we could assume that all bags simply contain N vectors of dimensionality D. Thus, we assume each $\gamma_{t,n} \in \mathbb{R}^D$ and we can rewrite each bag χ_t in an $N \times D$ matrix form as X_t. Our dataset of many bags can then be stored as T matrices and consists of $\{X_t\}_{t=1}^T$. To reorder each of these bags, we consider endowing each matrix X_t with an unknown $N \times N$ permutation matrix A_t which re-sorts its N row entries. Therefore, we augment our dataset with matrices that re-sort it as follows $\{A_t X_t\}_{t=1}^T$. In the more general case where we are not dealing with vectors for each $\gamma_{t,n}$, we will take the permutation matrices A_t to be a general permutation p_t of the set $\{1, \ldots, N\}$ which defines an ordering of the bag as follows $p_t \otimes \chi_t = \left(\gamma_{t,p_t(n)}\right)_{n=1}^N$. This gives us an ordered version of the dataset for a specific configuration of orderings denoted P which we write as follows $\mathcal{D}_P = \{p_t \otimes \chi_t\}_{t=1}^T$.

Given the original dataset, we want to find a *good* permutation configuration by optimizing the matrices $\{A_t\}_{t=1}^T$ or the permutation configurations $\{p_t\}_{t=1}^T$. To make the notion of goodness of permutation configurations concrete, we will argue that good permutations will reveal a compact low-dimensional representation of the data. For instance, the data may lie on a low dimensional manifold that is much smaller than the embedding space of size ND or $N|\gamma_{t,n}|$, where $|\gamma_{t,n}|$ is the dimensionality of the objects being permuted (if and when such a quantity makes sense). We now elaborate how to approximately measure the dimensionality of the potentially nonlinear manifold spanning the data. This is done by observing the eigenvalue spectrum of kernel PCA which approximates the volume data occupies in Hilbert space. Clearly, a low volume suggests that we are dealing with a low dimensional manifold in Hilbert space.

2.1 Kernel PCA and Gaussians in Hilbert space

We subscribe to the perspective that PCA finds a subspace from data by modeling it as a degenerate Gaussian since only first and second order statistics of a dataset $\{x_t\}_{t=1}^T$ are computed [7]. Similarly, kernel PCA finds a subspace in Hilbert space by only looking at first and second order statistics of the feature vectors $\{\phi(x_t)\}_{t=1}^T$ instead[1]. In fact, we are also restricted to second order statistics since we wish to use kernel methods and can thus only interact with data in Hilbert space via inner-products $k(x_t, x_{t'}) = \langle \phi(x_t), \phi(x_{t'}) \rangle$.

One way to evaluate the quality of a subspace discovered by kernel PCA is by estimating the volume occupied by the data. In cases where the volume of the data in Hilbert space is low, we anticipate that only a few kernel principal components will be necessary to span and reconstruct the dataset. Since kernel PCA hinges on Gaussian statistics, we will only use a second order estimator of the volume of our dataset. Consider computing the mean and covariance of a Gaussian from the dataset in Hilbert space. In kernel PCA [12], recall that the top eigenvalues of the covariance matrix $\Sigma = \frac{1}{T}\sum_t \phi(x_t)\phi(x_t)^T$ of the data are related to the top eigenvalues of the $T \times T$ Gram matrix K of the data which is defined element-wise as $[K]_{t,t'} = k(x_t, x_{t'})$. The eigenvalues λ and eigenvectors $\boldsymbol{\alpha}$ of the Gram matrix are given by the solution to the problem:

$$\begin{bmatrix} \langle k_{x_1}, k_{x_1} \rangle & \cdots & \langle k_{x_1}, k_{x_T} \rangle \\ \vdots & & \vdots \\ \langle k_{x_T}, k_{x_1} \rangle & \cdots & \langle k_{x_T}, k_{x_T} \rangle \end{bmatrix} \begin{bmatrix} \alpha_1 \\ \vdots \\ \alpha_T \end{bmatrix} = T\lambda \begin{bmatrix} \alpha_1 \\ \vdots \\ \alpha_T \end{bmatrix}.$$

From the above, we find the top J eigenvectors $\boldsymbol{\alpha}^j$ which produce the highest J eigenvalues and approximate the dataset with a J-dimensional nonlinear manifold. The eigenfunctions $v^j(x)$ of the covariance matrix describe axes of variation on the manifold and are unit-norm functions approximated by:

$$v^j(x) \propto \sum_{t=1}^{T} \alpha_t^j k(x, x_t).$$

These are normalized such that $\langle v^j, v^j \rangle = 1$. The spectrum of eigenvalues describes the overall shape of a Gaussian model of the data in Hilbert space while the eigenvectors of the covariance matrix capture the Gaussian's orientation. The volume of the data can then be approximated by the determinant of the covariance matrix which equals the product of its eigenvalues λ^j.

$$\text{Volume} \approx |\Sigma| = \prod_j \lambda^j.$$

If we are dealing with a truly low-dimensional subspace, only a few eigenvalues (corresponding to eigenvectors spanning the manifold) will be large. The many remaining eigenvalues corresponding to noise off of the manifold will be small and

[1] While this Hilbert space could potentially be infinite dimensional and Gaussians and kernel PCA should be handled more formally (i.e. using Gaussian processes with white noise and appropriate operators) in this paper and for our purposes we will assume we are manipulating only finite-dimensional Hilbert spaces. Formalising the extensions to infinite Hilbert space is straightforward.

the volume we ultimately estimate by multiplying all these eigenvalues will be low[2]. Thus, a kernel PCA manifold that is low-dimensional should typically have low volume. It is well known that kernel PCA can also be (implicitly) centered by estimating and removing the mean of the data yet we will not elaborate this straightforward issue (refer instead to [12]). Before applying PCA, recall that we perform maximum likelihood estimation to obtain the mean $\hat{\mu}$ and the covariance $\hat{\Sigma}$. The volume of the dataset is related to its log-likelihood under the maximum likelihood estimate of a Gaussian model as shown in [4]:

$$\begin{aligned} l(\mu, \Sigma) &= \sum_t \log \mathcal{N}(x_t|\mu, \Sigma) \\ &= -\frac{TD}{2}\log(2\pi) - \frac{T}{2}\log|\Sigma| - \frac{1}{2}\sum_t (x_t - \mu)^T \Sigma^{-1}(x_t - \mu). \end{aligned}$$

Log-likelihood simplifies as follows when we use the maximum likelihood setting for the mean $\hat{\mu} = \frac{1}{T}\sum_t x_t$ and covariance $\hat{\Sigma} = \frac{1}{T}\sum_t (x_t - \hat{\mu})(x_t - \hat{\mu})^T$.

$$l(\hat{\mu}, \hat{\Sigma}) = -\frac{TD}{2}\log(2\pi) - \frac{T}{2}\log|\hat{\Sigma}| - \frac{TD}{2}.$$

Therefore, we can see that a kernel PCA solution which has high log-likelihood according to the Gaussian mean and covariance will also have low volume low $\log|\hat{\Sigma}|$ and produce a compact low-dimensional manifold requiring few principal axis to span the data.

2.2 Permutations that maximize likelihood and minimize volume

We saw that we are solving a maximum likelihood problem to perform kernel PCA and higher likelihoods indicate lower volume and a better subspace. However, the above formulation assumes we have vectors or can readily compute kernels or inner products between kPCA's T Hilbert-space vectors $\{\phi(x_t)\}_{t=1}^T$. This is not trivial when each x_t is actually an unordered bag of tuples as we had when we were previously dealing with χ_t. However, given an ordering of each via A_t matrices or p_t permutations, we can consider computing a kernel on the sorted bags as follows:

$$k(p_t \otimes \chi_t, p_t \otimes \chi_{t'}) = \sum_{i=1}^N \left\langle \phi(\gamma_{t,p_t(i)}), \phi(\gamma_{t',p_{t'}(i)}) \right\rangle = \sum_{i=1}^N \kappa(\gamma_{t,p_t(i)}, \gamma_{t',p_{t'}(i)})$$

assuming we have defined a base kernel $\kappa(.,.)$ between the actual objects $\gamma_{t,n}$ in our bags. Another potentially clearer view of the above is to instead assume we have bags of Hilbert-space vectors where our dataset $\mathcal{D}$ has T of these sets or bags $\mathcal{D} = \{\Phi_t\}_{t=1}^T$. Each of these bags is merely a collection of N unordered

[2] Here we are assuming that we do not obtain any zero-valued eigenvalues which produce a degenerate estimate of volume. We will regularize eigenvalues in the subsequent sections to avoid this problem.

objects in Hilbert space $\Phi_t = \{\phi(\gamma_{t,n})\}_{n=1}^N$. Applying the ordering p_t to this unordered bag of Hilbert space vectors provides an ordered set as follows $p_t \otimes \Phi_t = \left(\phi(\gamma_{t,p_t(n)})\right)_{n=1}^N$. Inner products between two ordered bags are again given in terms of the base kernel $\kappa(.,.)$ as follows:

$$\langle p_t \otimes \Phi_t, p_t' \otimes \Phi_t' \rangle = \sum_{i=1}^N \left\langle \phi(\gamma_{t,p_t(i)}), \phi(\gamma_{t',p_{t'}(i)}) \right\rangle \;=\; \sum_{i=1}^N \kappa(\gamma_{t,p_t(i)}, \gamma_{t',p_{t'}(i)}).$$

As in [4] we will find settings of A_t or p_t that maximize likelihood under a Gaussian model to minimize volume. However, instead of directly minimizing the volume by assuming we always have updated the mean and covariance with their maximum likelihood setting, we will treat the problem as an iterative likelihood maximization scheme. We have the following log-likelihood problem which we argued measures the volume of the data at the maximum likelihood estimate of μ and Σ:

$$l(p_1, \ldots, p_T, \mu, \Sigma) = \sum_t l_t(p_t, \mu, \Sigma) \;=\; \sum_t \log \mathcal{N}(p_t \otimes \Phi_t | \mu, \Sigma).$$

Further increasing likelihood by adjusting $p_1, \ldots, p_T$ will also further decrease volume as we interleave updates of μ and Σ. Thus, the above is an objective function on permutations and maximizing it should produce an ordering of our bags that keeps kernel PCA efficient. Here, we are assuming we have a Gaussian in Hilbert space yet it is not immediately clear how to maximize or evaluate the above objective function and obtain permutation configurations that give low-volume kernel PCA manifolds. We will next elaborate this and show that all computations are straightforward to perform in Hilbert space.

We will maximize likelihood over $p_1, \ldots, p_T$, μ and Σ iteratively in an axis-parallel manner. This is done by locking all parameters of the log-likelihood and modifying a single one at a time. Note, first, that it is straightforward, given a current setting of $(p_1, \ldots, p_T)$ to compute the maximum likelihood μ and Σ as the mean and covariance in Hilbert space. Now, assume we have locked μ and Σ at a current setting and we wish to only increase likelihood by adjusting the permutation p_t of a single bag Φ_t. We investigate two separate cases. In the first case, we assume the covariance matrix Σ is locked at a scalar times identity and we find the optimal update for a given p_t by solving a linear assignment problem. We will then consider the more general case where the current Σ covariance matrix in Hilbert space is an arbitrary positive semi-definite matrix and updating the current p_t will involve solving a quadratic assignment problem.

3 Kernelized sorting via LAP and mean alignment

Given μ, $p_1, \ldots, p_T$ and $\Sigma = \sigma I$ we wish to find a setting of p_t which maximizes the likelihood of an isotropic Gaussian. This clearly involves only maximizing the following contribution of bag t to the total log-likelihood:

$$l_t(p_t, \mu, \Sigma) = \log \mathcal{N}(p_t \otimes \Phi_t | \mu, \sigma I).$$

We can simplify the above as follows:

$$l_t(p_t, \mu, \Sigma) = \text{const} - \frac{1}{2\sigma}\left(\langle p_t \otimes \Phi_t, p_t \otimes \Phi_t\rangle - 2\langle p_t \otimes \Phi_t, \mu\rangle + \langle \mu, \mu\rangle\right).$$

Since $\langle p_t \otimes \Phi_t, p_t \otimes \Phi_t\rangle$ is constant despite our choice of p_t, maximizing the above over p_t is equivalent to maximizing the following function:

$$\hat{p}_t = \arg\max_{p_t} \langle p_t \otimes \Phi_t, \mu\rangle .$$

Assume we have the current maximum likelihood mean which is computed from the locked permutation configurations from the previous iteration $\hat{p}_1, \ldots, \hat{p}_T$. The above then simplifies into:

$$\hat{p}_t = \arg\max_{p_t} \left\langle p_t \otimes \Phi_t, \frac{1}{T}\sum_{t'=1}^{T} \hat{p}_{t'} \otimes \Phi_{t'} \right\rangle \;=\; \arg\max_{p_t} \sum_{i=1}^{N}\sum_{t'=1}^{T} \kappa\left(\gamma_{t,p_t(i)}, \gamma_{t',\hat{p}_{t'}(i)}\right).$$

The above problem is an instance of the linear assignment problem (LAP) and can directly be solved producing the optimal p_t in $O(N^3)$ via the Kuhn-Munkres algorithm (or more efficient variants such as QuickMatch [10], auction algorithms or the cost scaling algorithm). Essentially, we find the permutation matrix A_t which is analogous to p_t by solving the assignment problem on the $N \times N$ matrix D_t via a simple call to the (standard) function KuhnMunkres($-D_t$) where D_t is an $N \times N$ matrix giving the value of kernel evaluations between items in the current bag and the mean bag. We define the D_t matrix element-wise as:

$$[D_t]_{i,i'} = \sum_{t'=1}^{T} \kappa\left(\gamma_{t,i}, \gamma_{t',\hat{p}_{t'}(i')}\right).$$

Iterating the update of each p_t in this way for $t = 1 \ldots T$ and updating the mean μ repeatedly by its maximum likelihood estimate will converge to a maximum of the log-likelihood. While a formal proof is deferred in this paper, this maximum may actually be global since the above problem is analogous to the generalized Procrustes problem [1]. In the general Procrustes setting, we can mimic the problem of aligning or permuting many bags towards a common mean by instead computing the alignments or permutations between all possible pairs of bags. For instance, it is possible to find permutations $p_{t,t'}$ or matrices $A_{t,t'}$ that align each bag χ_t to any other bag $\chi_{t'}$ via $[D_{t,t'}]_{i,i'} = \kappa(\gamma_{t,i}, \gamma_{t',i'})$. These then give a consistent set of permutations to align the data towards a common mean prior to kernel PCA. This provides us with the ordering $\hat{p}_1, \ldots, \hat{p}_T$ of the data which now becomes a dataset of ordered bags $\{\hat{p}_t \otimes \Phi_t\}_{t=1}^{T}$. Subsequently, we perform kernel PCA on the data in $O(T^3)$ using singular value decomposition on the $T \times T$ centered Gram matrix. This gives the eigenvectors, eigenvalues and eigenfunctions that span the nonlinear manifold representation of the ordered data. This will have a higher likelihood and potentially use fewer principal components to achieve the same reconstruction accuracy than immediate application of kernel PCA on the dataset $\mathcal{D}$. Of course, this argument only holds if the dataset itself truly has a natural permutation invariance or was a collection of sets or bags.

We now turn to the more general case where the Gaussian covariance is arbitrary and is not artificially locked at a spherical configuration. However, in this setting, global convergence claims are even more elusive.

4 Kernelized sorting via QAP and covariance alignment

In the case where we consider anisotropic Gaussians, the covariance matrix is an arbitrary positive semi-definite matrix and we have a more involved procedure for updating a given p_t. However, this is more closely matched to the full problem of minimizing the volume of the data and should produce more valuable orderings that further reduce the number of kernel principal components we need to represent the ordered bags. Here, we are updating a single p_t again yet the covariance matrix Σ is not a scaled identity. We therefore have the following contribution of bag t to the log-likelihood objective function:

$$l_t(p_t, \mu, \Sigma) = \log \mathcal{N}(p_t \otimes \Phi_t | \mu, \Sigma).$$

Due to the presence of the Σ, this will no longer reduce to a simple linear assignment problem that is directly solvable for A_t or $\hat{p}_t$ using a polynomial time algorithm. In fact, this objective will produce an NP-Complete quadratic assignment problem [9]. Instead we will describe an iterative technique for maximizing the likelihood over p_t by using a variational upper bound on the objective function.

Define the inverse matrix $M = \Sigma^{-1}$ which we will assume has actually been regularized as follows $M = (\Sigma + \epsilon_1 I)^{-1} + \epsilon_2 I$ where ϵ_1 and ϵ_2 are small scalars (the intuition for this regularization is given in [5]). Recall kernel PCA (with abuse of notation) gives the matrix Σ as follows $\Sigma = \sum_j \lambda^j v^j (v^j)^T$. Meanwhile, the matrix M can also be expressed with abuse of notation in terms of its eigenvalues $\tilde{\lambda}_k$ and eigenfunctions v^j from as follows $M = \sum_{j=1}^{J} \tilde{\lambda}^j v^j (v^j)^T + \sigma I$. We can assume we pick a finite J that is sufficiently large to have a faithful approximation to M. Recall that, as in kernel PCA, the (unnormalized) eigenfunctions are given by the previous estimate of the inverse covariance at the previous (locked) estimates of the permutations $\hat{p}_t$:

$$\left\langle v^j, p \otimes \Phi \right\rangle = \sum_{t=1}^{T} \alpha_t^j \left\langle p \otimes \Phi, \hat{p}_t \otimes \Phi_t \right\rangle$$

where the normalization such that $\left\langle v^j, v^j \right\rangle = 1$ is absorbed into the $\tilde{\lambda}^j$ for brevity. We can now rewrite the (slightly regularized) log-likelihood more succinctly by noting that μ and Σ are locked (thus some terms become mere constants):

$$\begin{aligned}
l_t(p_t) &= \text{const} - \frac{1}{2}(p_t \otimes \Phi_t - \mu)^T M (p_t \otimes \Phi_t - \mu) \\
&= \text{const} - \frac{1}{2}(p_t \otimes \Phi_t)^T M (p_t \otimes \Phi_t) + (p_t \otimes \Phi_t)^T M \mu \\
&= \text{const} - \frac{1}{2}(p_t \otimes \Phi_t)^T \sum_{j=1}^{J} \tilde{\lambda}^j v^j (v^j)^T (p_t \otimes \Phi_t) + (p_t \otimes \Phi_t)^T M \mu
\end{aligned}$$

where we have used the expanded definition of the M matrix yet its isotropic contribution σI as before has no effect on the quadratic term involving p_t. However, the anisotropic contribution remains and we have a QAP problem which we continue simplifying by writing the eigenvectors as linear combinations of Hilbert space vectors or kernel functions:

$$\begin{aligned}
l_t(p_t) = \text{const} &- \frac{1}{2}\sum_{j=1}^{J} \tilde{\lambda}^j \left(\sum_{m=1}^{T} \alpha_m^j \left\langle p_t \otimes \Phi_t, \hat{p}_m \otimes \Phi_m \right\rangle \right)^2 \\
&+ \sum_{j=1}^{J} \tilde{\lambda}^j \sum_{m=1}^{T} \alpha_m^j \left\langle p_t \otimes \Phi_t, \hat{p}_m \otimes \Phi_m \right\rangle \sum_{n=1}^{T} \alpha_n^j \left\langle \mu, \hat{p}_n \otimes \Phi_n \right\rangle + \sigma \left\langle p_t \otimes \Phi_t, \mu \right\rangle .
\end{aligned}$$

For notational convenience, exchange the p_t notation and start using the permutation matrix notation A_t by noting the following relationship:

$$\left\langle p_t \otimes \Phi_t, \hat{p}_{t'} \otimes \Phi_{t'} \right\rangle = \sum_{i=1}^{N} \sum_{i'=1}^{N} [A_t]_{i,i'} \kappa(\gamma_{t,i}, \gamma_{t',\hat{p}_{t'}(i')}).$$

We can now rewrite the (negated) log-likelihood term as a cost function $C(A_t) \equiv -l(A_t)$ over the space of permutation matrices A_t. This cost function is as follows after we drop some trivial constant terms:

$$C(A_t) = \sum_{j=1}^{J} \frac{\tilde{\lambda}^j}{2} \left(\sum_{i=1}^{N} \sum_{i'=1}^{N} [A_t]_{i,i'} \sum_{m=1}^{T} \alpha_m^j \kappa(\gamma_{t,i}, \gamma_{m,\hat{p}_m(i')}) \right)^2 - \sum_{i=1}^{N} \sum_{i'=1}^{N} [A_t]_{i,i'} [D_t]_{i,i'}$$

where we have defined the readily computable $N \times N$ matrix D_t element-wise as follows for brevity:

$$\begin{aligned}
[D_t]_{i,i'} = &\sum_{j=1}^{J} \tilde{\lambda}^j \sum_{m=1}^{T} \alpha_m^j \kappa(\gamma_{t,i}, \gamma_{m,\hat{p}_m(i')}) \left(\sum_{n=1}^{T} \alpha_n^j \left\langle \mu, \hat{p}_n \otimes \Phi_n \right\rangle \right) \\
&+ \frac{\sigma}{T} \sum_{t'=1}^{T} \kappa(\gamma_{t,i}, \gamma_{t',\hat{p}'_t(i')}).
\end{aligned}$$

This matrix degenerates to the previous isotropic case if all anisotropic Lagrange multipliers go to zero leaving only the σI contribution. Note, we can fill in the terms in the parentheses as follows:

$$\langle \mu, \hat{p}_n \otimes \Phi_n \rangle = \frac{1}{T}\sum_{t'=1}^{T} \langle \hat{p}_{t'} \otimes \Phi_{t'}, \hat{p}_n \otimes \Phi_n \rangle \;=\; \frac{1}{T}\sum_{t'=1}^{T}\sum_{i=1}^{N} \kappa(\gamma_{n,\hat{p}_n(i)}, \gamma_{t',\hat{p}_{t'}(i)})$$

which lets us numerically compute the D_t matrix's $N \times N$ entries.

Clearly the first term in $C(A_t)$ is quadratic in the permutation matrix A_t while the second term in $C(A_t)$ is linear in the permutation matrix. Therefore, the second LAP term could be optimized using a Kuhn-Munkres algorithm however, the full cost function is a quadratic assignment problem. To address this issue, we will upper bound the first quadratic cost term with a linear term such that we can minimize $C(A_t)$ iteratively using repeated applications of Kuhn-Munkres. This approach to solving QAP iteratively via bounding and LAP is similar in spirit to the well-known Gilmore-Lawler bound method as well as other techniques in the literature [9].

First, we construct an upper bound on the cost by introducing two $J \times N$ matrices called Q and $\tilde{Q}$. The entries of both Q and $\tilde{Q}$ are non-negative and have the property that summing across their columns gives unity as follows:

$$\sum_i [Q]_{j,i} \;=\; 1 \quad \text{and} \quad \sum_{i'} [\tilde{Q}]_{j,i'} \;=\; 1.$$

We insert the ratio of a convex combination of these two matrices (weighted by a positive scalar $\delta^j \in [0,1]$) into our cost such that $C(A_t) =$

$$\sum_{j=1}^{J} \frac{\tilde{\lambda}^j}{2} \left(\sum_{i=1}^{N}\sum_{i'=1}^{N} [A_t]_{i,i'} \frac{\delta^j [Q]_{j,i} + (1-\delta^j)[\tilde{Q}]_{j,i'}}{\delta^j [Q]_{j,i} + (1-\delta^j)[\tilde{Q}]_{j,i'}} \sum_{m=1}^{T} \alpha_m^j \kappa(\gamma_{t,i}, \gamma_{m,\hat{p}_m(i')}) \right)^2 - \sum_{i=1}^{N}\sum_{i'=1}^{N} [A_t]_{i,i'} [D_t]_{i,i'}.$$

Note that this in no way changes the cost function, we are merely multiplying each entry of the matrix A_t by unity. Next recall that the squaring function $f(x) = x^2$ is convex and we can therefore apply Jensen's inequality to pull terms out of it. We first recognize that we have a convex combination within the squaring since:

$$\sum_{i=1}^{N}\sum_{i'=1}^{N} [A_t]_{i,i'} \left(\delta^j [Q]_{j,i} + (1-\delta^j)[\tilde{Q}]_{j,i'} \right) = \delta^j + (1-\delta^j) \;=\; 1 \quad \forall\, j.$$

Therefore, we can proceed with Jensen to obtain the upper bound on cost as follows, $C(A_t) \leq$

$$\sum_{j=1}^{J} \frac{\tilde{\lambda}^j}{2} \sum_{i=1}^{N}\sum_{i'=1}^{N} [A_t]_{i,i'} \left(\delta^j [Q]_{j,i} + (1-\delta^j)[\tilde{Q}]_{j,i'} \right) \left(\frac{\sum_{m=1}^{T} \alpha_m^j \kappa(\gamma_{t,i}, \gamma_{m,\hat{p}_m(i')})}{\delta^j [Q]_{j,i} + (1-\delta^j)[\tilde{Q}]_{j,i'}} \right)^2 - \sum_{i=1}^{N}\sum_{i'=1}^{N} [A_t]_{i,i'} [D_t]_{i,i'}.$$

The above bound is actually just a linear assignment problem (LAP) which we write succinctly as follows:

$$C(A_t) \leq \sum_{i=1}^{N} \sum_{i'=1}^{N} [A_t]_{i,i'} \left[\sum_{j=1}^{J} \frac{\tilde{\lambda}^j}{2} \frac{\left(\sum_{m=1}^{T} \alpha_m^j \kappa(\gamma_{t,i}, \gamma_{m,\hat{p}_m(i')}) \right)^2}{\delta^j [Q]_{j,i} + (1-\delta^j)[\tilde{Q}]_{j,i'}} - [D_t]_{i,i'} \right].$$

The above upper bound can immediately be minimized over permutation matrices and gives A_t via a Kuhn-Munkres computation or some variant. However, we would need to actually specify Q, $\tilde{Q}$ and all the δ^j for this computation. In fact, the right hand side is a variational LAP bound over our original QAP with the (augmented parameters) over Q, $\tilde{Q}$, $\boldsymbol{\delta} = (\delta^1, \ldots, \delta^J)$ and A_t which can each be iteratively minimized. Thus, we anticipate repeatedly minimizing over A_t using Kuhn-Munkres operations followed by updates of the remaining bound parameters given a current setting of A_t. Note, the left term in the square bracket is constant if all eigenvalues $\tilde{\lambda}_j$ are equal (in which case the log-likelihood term overall is merely an LAP). Thus, we can see that the variance in the eigenvalues is likely to have some effect as we depart from a pure LAP setting to a more *severe* QAP setting. This variance in eigenvalue spectrum can give us some indication about the convergence of the iterative procedure.

We next minimizing the bound on the right hand size over Q and $\tilde{Q}$ which is written more succinctly as follows:

$$\min_{Q} \min_{\tilde{Q}} \sum_{i=1}^{N} \sum_{i'=1}^{N} \sum_{j=1}^{J} \frac{[P^j]_{i,i'}}{\delta^j [Q]_{j,i} + (1-\delta^j)[\tilde{Q}]_{j,i'}}$$

where we have defined each matrix P^j element-wise using the formula at the current setting of A_t

$$[P^j]_{i,i'} = [A_t]_{i,i'} \tilde{\lambda}^j \left(\sum_{m=1}^{T} \alpha_m^j \kappa(\gamma_{t,i}, \gamma_{m,\hat{p}_m(i')}) \right)^2 .$$

This is still not directly solvable as is. Therefore we consider another variational bounding step (which leads to more iterations) by applying Jensen on the convex function $f(x) = 1/x$ (this is true only when x is non-negative which is the case here). This produces the following inequality:

$$\sum_{i=1}^{N} \sum_{i'=1}^{N} \sum_{j=1}^{J} \frac{[P^j]_{i,i'}}{\delta^j [Q]_{j,i} + (1-\delta^j)[\tilde{Q}]_{j,i'}} \leq \sum_{i=1}^{N} \sum_{i'=1}^{N} \sum_{j=1}^{J} \delta^j \frac{[P^j]_{i,i'}}{[Q]_{j,i}} + (1-\delta^j) \frac{[P^j]_{i,i'}}{[\tilde{Q}]_{j,i'}}$$

Clearly, once we have invoked the second application of Jensen's inequality on this function, we get an easy update rule for Q by taking derivatives and setting to zero. In addition, we introduce the Lagrangian constraint that enforces the summation to unity $\sum_i [Q]_{j,i} = 1$. Ultimately, we obtain this update rule:

$$[Q]_{j,i} = \frac{\sum_{i'} \sqrt{[P^j]_{i,i'}}}{\sum_{i,i'} \sqrt{[P^j]_{i,i'}}}.$$

Similarly, $\tilde{Q}$ is updated as follows:

$$[\tilde{Q}]_{j,i'} = \frac{\sum_i \sqrt{[P^j]_{i,i'}}}{\sum_{i,i'} \sqrt{[P^j]_{i,i'}}}.$$

The remaining update rule for the δ^j values is then given as follows:

$$\min_{\boldsymbol{\delta}} \sum_{i=1}^{N} \sum_{i'=1}^{N} \sum_{j=1}^{J} \frac{[P^j]_{i,i'}}{\delta^j [Q]_{j,i} + (1-\delta^j)[\tilde{Q}]_{j,i'}}$$

The terms for each single δ^j are independent and yield the following:

$$\min_{\delta^j} \sum_{i=1}^{N} \sum_{i'=1}^{N} \frac{[P^j]_{i,i'}}{\delta^j [Q]_{j,i} + (1-\delta^j)[\tilde{Q}]_{j,i'}}$$

One straightforward manner to minimize the above extremely simple cost over a scalar $\delta^j \in [0,1]$ is to use brute force techniques or bisection/Brent's search.

Thus, we can iterate updates of Q, $\tilde{Q}$, and the $\boldsymbol{\delta}$ with updates of A_t to iteratively minimize the upper bound on $C(A_t)$ and maximize likelihood. Updating A_t is straightforward via a Kuhn Munkres algorithm (or faster heuristic algorithms such as QuickMatch [10]) on the terms in the square bracket multiplying the entries of the A_t matrix (in other words, iterate a linear assignment problem, LAP). Convergence of this iterative scheme is reasonable and improves the likelihood as we update A_t. But, it may have local minima[3]. We are working on even tighter bounds that seem promising and should further improve convergence and alleviate the local minima problem. Once the iterative scheme converges for a given bag Φ_t, we obtain the A_t matrix which directly gives the permutation configuration p_t.

We continue updating the p_t for each bag in our data set while also updating the mean and the covariance (or, equivalently, the eigenvalues, eigenvectors and eigenfunctions for kernel PCA). This iteratively maximizes the log-likelihood (and minimizes the volume of the data) until we reach a local maximum and converge to a final ordering of our dataset of bags $\{\hat{p}_t \otimes \Phi_t\}_{t=1}^{T}$.

5 Implementation Details

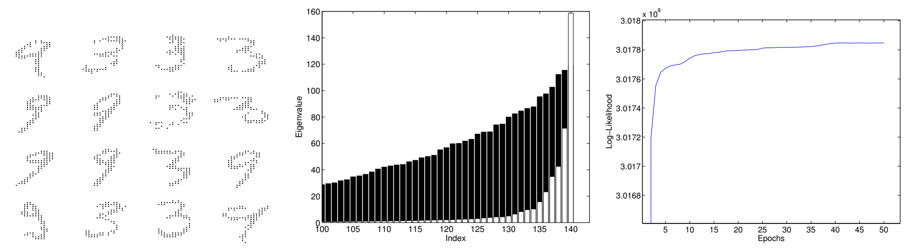
We now discuss some particular implementation details of applying the method in practice. First, we are not bound to assuming that there must be exactly N objects in each bag. Assume we are given $t = 1 \ldots T$ bags with a variable number N_t of objects in each bag. We first pick a constant N (typically $N = \max_t N_t$) and then randomly replicate (or sample without replacement for small N) the objects in each bag such that each bag has N objects. Another consideration is that we generally hold the permutation of one bag fixed since permutations are

[3] This is not surprising since QAP is NP-Complete.

relative. Therefore, the permutation p_1 for bag Φ_1 is locked (i.e. for a permutation matrix we would set $A_1 = I$) and only the remaining permutations need to be optimized. We then iterate through the data randomly updating each p_t at a time from the permutations $p_2, \ldots, p_T$. We first start by using the mean estimator (LAP) and update its estimate for each p_t until it longer reduces the volume (as measured by the regularized product of kPCA's eigenvalues). We then iterate the update rule for the covariance QAP estimator until it no longer reduces volume. Finally, once converged, we perform kernel PCA on the sorted bags with the final setting of $p_2, \ldots, p_T$.

6 Experiments

In a preliminary experiment, we obtained a dataset of $T = 100$ digits of 9's and 3's as shown in Figure 2(a). Each digit is actually a bag or a set of $N = 70$ total (x, y) coordinates which form our $\gamma_{t,n} \in \mathbb{R}^2$. We computed the optimal permutations p_t for each digit using the minimum volume criterion (i.e. maximum likelihood with the anisotropic Gaussian case). Figure 2(b) shows the eigenvalue spectrum for PCA before ordering (i.e. assuming the given pseudo-random ordering in the raw input dataset) as well as the eigenvalue spectrum after optimizing the ordering. Note that lower eigenvalues indicate a smaller subspace and that there are few true dimensions of variability in the data once we sort the bags.



(a) Bags of points dataset (b) Eigenvalue spectrum (c) Log-Likelihood

Fig. 2. Ordering figits as bags of permutable point-clouds prior to PCA. In (a) we see a sample of the original training set of 100 digits while in (b) we see the original PCA eigenvalue spectrum (darker bars) with the initial pseudo-random ordering in the data. In (b) we see the eigenvalue spectrum (lighter bars) after optimizing the ordering to minimize the volume of the subspace (or maximize likelihood under an anisotropic Gaussian). In (c), note the increasing log-likelihood as we optimize each $\hat{p}_t$.

To visualize the resulting orderings, we computed linear interpolations between the sorted bags for different pairs of digits in the input dataset. Figure 3 depicts the morphing as we mix the coordinates of each dot in each digit with another. Note in (a), these 'bags of coordinates' are unordered. Therefore, blending their coordinates results in a meaningless cloud of points during the transition.

However, in (b), we note that the points in each bag or cloud are corresponded and ordered so morphing or linearly interpolating their coordinates for two different digits results in a meaningful smooth movement and bending of the digit. Note that in (b) morphs from 3 to another 3, 9 to another 9 or a 3 to a 9 maintain meaningful structure at the half-way point as we blend between one digit and another. This indicates a more meaningful ordering has emerged unlike the initial random one which, when blending between two digit shapes, always generates a random cloud of (x, y) coordinates (see Figure 3(a)). For this dataset, results were similar for the mean vs. covariance estimator as well as linear vs. quadratic choices for the base kernel $\kappa(.,.)$.

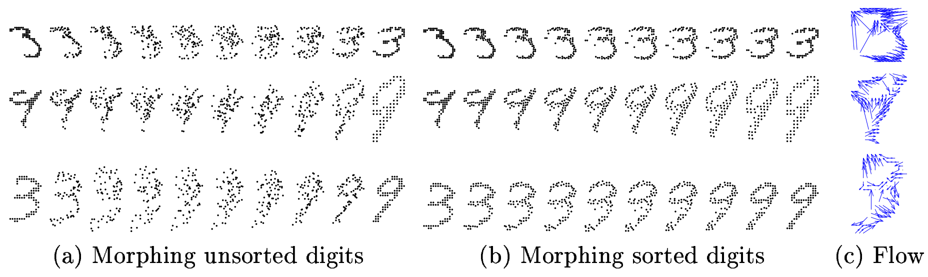

(a) Morphing unsorted digits (b) Morphing sorted digits (c) Flow

Fig. 3. Linear interpolation from left to right (morphing) of the point-clouds with and without sorting. In (a) we see the linear morphing between unordered point clouds which results in poor intermediate morphs that are not meaningful. Meanwhile in (b) where we have recovered good orderings p_t for each digit by minimizing the Gaussian's volume, we note that the digits preserve the correspondence between different parts and induce a smooth and natural morph between the two initial digit configurations. In (c) we show the two digits with arrows indicating the flow or correspondence.

7 Conclusions

We have proposed an algorithm for finding orderings or sortings of multiple sets of objects. These sets or bags need not contain scalars or vectors but rather contain N arbitrary objects. Interacting with these objects is done solely via kernel functions on pairs of them leading to a general notion of sorting in Hilbert space. The ordering or sorting we propose is such that we form a low-dimensional kernel PCA approximation with as few eigenfunctions as possible to reconstruct the manifold on which these bags exist. This is done by finding the permutations of the bags such that we move them towards a common mean in Hilbert space or a low-volume Gaussian configuration in Hilbert space. In this article, this criterion suggested two maximum likelihood objective functions: one which is a linear assignment problem and the other a quadratic assignment problem. Both can be iteratively minimized by using a Kuhn Munkres algorithm along with variational bounding. This permits us to sort or order sets in a general way in Hilbert space using kernel methods and to ultimately obtain a compact representation of the data. We are currently investigating ambitious applications of the method with various kernels and additional results available at:

`http://www.cs.columbia.edu/~jebara/bags/`

In future work, we plan on investigating discriminative variations of the sorting/ordering problem to build classifiers based on support vector machines or kernelized Fisher discriminants that sort data prior to classification (see [4] which elaborates a quadratic cost function for the Fisher discriminant).

Acknowledgments

Thanks to R. Dovgard, R. Kondor and the reviewers for suggestions. T. Jebara is supported in part by NSF grants CCR-0312690 and IIS-0347499.

References

1. Ian L. Dryden and Kanti V. Mardia. *Statistical Shape Analysis.* John Wiley and Sons, 1998.
2. S. Gold, C.P. Lu, A. Rangarajan, S. Pappu, and E. Mjolsness. New algorithms for 2D and 3D point matching: Pose estimation and correspondence. In *NIPS 7*, 1995.
3. A.V. Goldberg and R. Kennedy. An efficient cost scaling algorithm for the assignment problem. *Mathematical Programming*, 71(2):153–178, 1995.
4. T. Jebara. Convex invariance learning. In *9th International Workshop on Artificial Intelligence and Statistics*, 2003.
5. T. Jebara. Images as bags of pixels. In *International Conference on Computer Vision*, 2003.
6. S. Kirshner, S. Parise, and P. Smyth. Unsupervised learning with permuted data. In *Machine Learning: Tenth International Conference, ICML*, 2003.
7. R. Kondor and T. Jebara. A kernel between sets of vectors. In *Machine Learning: Tenth International Conference, ICML*, 2003.
8. J. Kosowsky and A. Yuille. The invisible hand algorithm: Solving the assignment problem with statistical physics. *Neural Networks*, 7:477–490, 1994.
9. Y. Li, P. M. Pardalos, K. G. Ramakrishnan, and M. G. C. Resende. Lower bounds for the quadratic assignment problem. *Annals of Operations Research*, 50:387–411, 1994.
10. J.B. Orlin and Y. Lee. Quickmatch: A very fast algorithm for the assignment problem. Technical Report WP# 3547-93, Sloan School of Management, Massachusetts Institute of Technology, March 1993.
11. Bernhard Schölkopf and Alexander J. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization and Beyond.* MIT Press, 2001.
12. Bernhard Schölkopf, Alexander J. Smola, and K.-R. Müller. Nonlinear principal component analysis as a kernel eigenvalue problem. *Neural Computation*, 10:1299–1319, 1998.
13. P. Y. Simard, Y. LeCun, J. S. Denker, and B. Victorri. Transformation invariance in pattern recognition – tangent distance and tangent propagation. *International Journal of Imaging Systems and Technology*, 11(3), 2000.
14. J.B. Tenenbaum and W.T. Freeman. Separating style and content with bilinear models. *Neural Computation*, 12(6), 1999.