

On adding and subtracting eigenspaces with EVD and SVD

Peter Hall^a David Marshall^b Ralph Martin^b

^a*School of Mathematical Sciences, Bath University*

^b*Department of Computer Science, Cardiff University*

Abstract

This paper provides two algorithms: one for adding eigenspaces, another for subtracting them, thus allowing for incremental updating and downdating of data models. Importantly, and unlike previous work, we keep an accurate track of the mean of the data, which allows our methods to be used in classification applications. The result of adding eigenspaces, each made from a set of data, is an approximation to that which would obtain were the sets of data taken together. Subtracting eigenspaces yields a result approximating that which would obtain were a subset of data used. Using our algorithms it is possible to perform “arithmetic” on eigenspaces without reference to the original data. We illustrate the use of our algorithms in three generic applications, including the dynamic construction of Gaussian mixture models. In addition, we mention singular value decomposition as an alternative to eigenvalue decomposition. We show that updating SVD models comes at the cost of space resources, and argue that downdating SVD models is not possible in closed-form.

Key words:

Eigenvalue decomposition, dynamic updating and downdating, Gaussian mixture models, singular value decomposition.

1 Introduction

This subject of this paper is incremental eigenanalysis: we provide an algorithm for including new data into an eigenspace, and another for removing data. An eigenspace comprises: the number of data points, their mean, the eigenvectors, and the eigenvalues that result from the eigenvalue decomposition (EVD) of the data covariance matrix. Typically the eigenspace is *deflated*, which is to say that only “significant” eigenvectors and eigenvalues are

retained in the eigenspace. The inclusion of new data is sometimes called *updating*, while the removal of data is sometimes called *downdating*. Rather than use data directly, we use eigenspace representations of the data, hence we *add* or *subtract* eigenspaces. Our methods are presented in Section 2.

Singular value decomposition (SVD) is closely related to EVD, and is preferred by some authors. We comment upon SVD methods in Section 4, where we present a novel method for incrementally computing SVD, including a shift of the mean. We also argue that removing data from SVD is not possible in closed form.

We must make clear the difference between *batch* and *incremental* methods for computing eigenspace models. A batch method computes an eigenmodel using all observations simultaneously. An incremental method computes an eigenspace model by successively updating an earlier model as new observations become available. In either case, the observations used to construct the eigenspace model are the *training observations*; that is, they are assumed to be instances from some class. This model may then be used to decide whether further observations belong to the class.

Incremental eigenanalysis has been studied previously [1–4,7,13], but surprisingly these authors either have ignored the fact that a change in data changes the mean, or else have handled it in an *ad hoc* way. In contrast, our previous work allows for a change of mean [10], where we allowed for the inclusion of only a *single* new datum. Here, our algorithms handle *block* update and down-date, so many observations can be included or removed in a single step. They explicitly allow the mean to vary in a principled and accurate manner, and this is important. Consider, for example, that functions such as the Mahalanobis distance, often used in classification applications, cannot be computed without the mean; previous solutions cannot be used in this case.

Applications of incremental methods are wide ranging both within computer vision and beyond. Focusing on computer vision, applications include: face recognition [12], modelling variances in geometry [6], and the estimation of motion parameters [4].

Our motivations for this work arose from several sources, one example being the construction of classification models for many images — too many to fit all into memory at once. Intuition, confirmed by experiment, suggests it is better to construct the eigenspace model from all the images rather than a subset of them, which is all that could be done if using a batch method; hence the need for an incremental method (see Section 3). An example is a database of photographs for a security application in which images need to be added and deleted each year, yet not all images can be kept in memory at once (see Section 3). Our methods allow the database to be updated and downdated

without recomputing the eigenmodel *ab initio*.

We are also interested in constructing dynamic Gaussian-mixture-models (GMMs), that is being able to add and subtract GMMs. For this, the ability to keep track of the mean while adding (or subtracting) eigenspaces is essential. A full discussion of the issues involved is beyond the scope of the paper, and is the subject of future work, but we present initial results (see Section 3) because of the potential of dynamic GMMs. For example, the mixture model used by Cootes and Taylor [5] can be brought into a dynamic learning framework, and since our GMMs rely on a hierarchy of subspaces, so too can work such as that of Heap and Hogg [11].

2 Adding and subtracting eigenspaces

We now state the problems which are our subject.

Let $X = [x_1, \dots, x_N]$ be a collection of N data points, each n dimensional. The eigenmodel of these data is

$$\Omega(X) = (\mu(X), U(X)_{np}, \Lambda(X)_p, N(X)) \quad (1)$$

in which: $\mu(X)$ is their mean; $U(X)_{np}$ is a collection of p eigenvectors, each a column in an $n \times p$ matrix; $\Lambda(X)_p$ is a collection of p eigenvalues, one per eigenvector, and N is the number of data points. The subscripts on each element identify its size, where we deem it helpful.

Typically $p \leq \min(n, N)$ is the rank of the covariance matrix of X , though this depends on details of how the model is deflated (see our previous work [10] for a discussion). We call p the dimension of the eigenspace. We have that $U^T U = I$, but usually $U U^T \neq I$, so the eigenvectors support a subspace of dimension p in a space of dimension n . The eigenvalues measure the standard deviation of the data over each of the eigenvectors, under the assumption that the data are Gaussian distributed. Hence, $\Omega(X)$ may be regarded as representing a multidimensional Gaussian distribution over a hyperplane, of dimension p , in some embedding space, of dimension n . Contours of equal likelihood generate hyperellipses of dimension p .

Another collection of observations $Y = [y_1, \dots, y_M]$ has an eigenmodel

$$\Omega(Y) = (\mu(Y), U(Y)_{nq}, \Lambda(Y)_q, N(Y)) \quad (2)$$

This collection is usually distinct from X , but such distinction is not a requirement. Notice that q eigenvectors and eigenvalues are kept in this model,

and in general $q \neq p$ even if $Y = X$: deflation may occur in different ways.

The problem for addition is to compute the eigenmodel for the pair of collections $Z = [X, Y]$

$$\begin{aligned}\Omega(Z) &= (\mu(Z), U(Z)_{nr}, \Lambda(Z)_r, N(Z)) & (3) \\ &= \Omega(X) \oplus \Omega(Y) & (4)\end{aligned}$$

with reference to $\Omega(X)$ and $\Omega(Y)$ only; that is, define the algorithm for the \oplus operator. *We assume the original data are not available.* In general, the number of eigenvectors and eigenvalues kept, r , differs from both p and q . This implies that addition must account for a possible change in dimension of the eigenspace.

The problem for subtraction is to compute $\Omega(X)$

$$\Omega(X) = \Omega(Z) \ominus \Omega(Y) \quad (5)$$

which is to remove the observations in Y from the eigenmodel in Z . As in the case of addition, a possible change in the dimension of the eigenspace must be accounted for.

This paper has space only to present the solutions to these problems, and derivations are available elsewhere [9].

2.1 Addition

Incremental computation of $N(Z)$ and $\mu(Z)$ is straightforward:

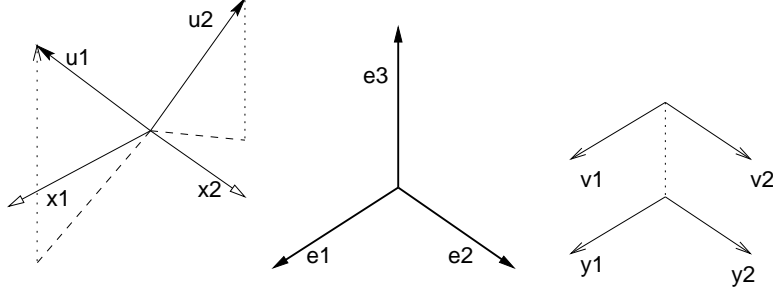
$$\begin{aligned}N(Z) &= N(X) + N(Y) & (6) \\ \mu(Z) &= (N(X)\mu(X) + N(Y)\mu(Y))/N(Z) & (7)\end{aligned}$$

Computing eigenvectors and eigenvalues depends upon properties of the subspaces that the eigenvectors $U(X)$, $U(Y)$, and $U(Z)$ support; properties we describe next.

Since $U(Z)$ must support all data in both collections, X and Y , both $U(X)$ and $U(Y)$ must be subspaces of $U(Z)$. Generally, we might expect that these subspaces “intersect” in the sense that $U(X)^T U(Y) \neq 0$. The null space of each of $U(X)$ and $U(Y)$ may contain some component of the other, that is to say $H = U(Y) - U(X)(U(X)^T U(Y)) \neq 0$. Both of these conditions are illustrated in Figure 1. Furthermore, even if $U(X)$ and $U(Y)$ are each a basis

for the same subspace, $U(Z)$ could be of larger dimension. This is because some component, h say, of the vector joining the means, $\mu(X) - \mu(Y)$ may be in the null space of both subspaces, simultaneously. For example $\mu(X)$, $U(X)$ and $\mu(Y)$, $U(Y)$ define a pair of planes parallel to the xy -plane, but separated in the z direction, as in Figure 1.

Subspaces $(x1,x2)$ and $(u1,u2)$ are embedded in $(e1,e2,e3)$
 Subspace $(u1,u2)$ has components in $(x1,x2)$, marked by dashed lines.
 It also has components in the null space of $(x1,x2)$, marked by dotted lines.
 Each component is embedded in $(e1,e2,e3)$.



Subspaces $(y1,y2)$ and $(v1,v2)$ are embedded in $(e1,e2,e3)$.
 They are parallel subspaces, the line joining them is in the null space of both.
 This line counts as an extra dimension when adding eigenspaces.

Fig. 1. An illustration of relationships between subspaces embedded in a larger space: intersecting subspaces (left); and parallel subspaces (right).

Putting (temporarily) to one side issues relating to changes in dimension, adding data acts to rotate the eigenvectors and scale the eigenvalues. Hence, the new eigenvectors must be a linear combination of the old. We deal with a change in dimension by constructing a basis sufficient to span $U(Z)$, for which we use $U(X)$, and a basis, ν that spans $[H, h]$, which is in the null space of $U(X)$ with respect to $U(Z)$. We have $U(Z) = [U(X), \nu]R$ where R is an orthonormal (rotation) matrix to be found by solving the following eigenproblem:

$$\begin{aligned} & \frac{N(X)}{N(Z)} \begin{bmatrix} \Lambda(X)_{pp} & 0_{pt} \\ 0_{tp} & 0_{tt} \end{bmatrix} + \\ & \frac{N(Y)}{N(Z)} \begin{bmatrix} G_{pq}\Lambda(Y)_{qq}G_{pq}^T & G_{pq}\Lambda(Y)_{qq}\Gamma_{tq}^T \\ \Gamma_{tq}\Lambda(Y)_{qq}G_{pq}^T & \Gamma_{tq}\Lambda(Y)_{qq}\Gamma_{tq}^T \end{bmatrix} + \\ & \frac{N(X)N(Y)}{N(Z)^2} \begin{bmatrix} g_p g_p^T & g_p \gamma_t^T \\ \gamma_t g_p^T & \gamma_t \gamma_t^T \end{bmatrix} = R_{ss} \Pi_{ss} R_{ss}^T \end{aligned} \quad (8)$$

in which Π is diagonal and

$$g_p = U(X)^T(\mu(X) - \mu(Y)) \quad (9)$$

$$G_{pq} = U(X)^T U(Y) \quad (10)$$

$$H_{nq} = [U(Y) - U(X)G_{pq}] \quad (11)$$

$$h_n = (\mu(X) - \mu(Y)) - U(X)g_p \quad (12)$$

$$\nu_{nt} = \text{Orthobasis}(\zeta[H_{nq}, h_n]) \quad (13)$$

$$\Gamma_{tq} = \nu_{nt}^T U(Y)_{nq} \quad (14)$$

$$\gamma_t = \nu^T(\mu(X) - \mu(Y)) \quad (15)$$

ζ is an operation that removes very small column vectors from the matrix, and **Orthobasis** computes a set of mutually orthogonal, unit vectors that support its argument; typically Gram-Schmidt orthogonalisation [8] is used to compute significant support vectors, ν from $\zeta[H, h]$; these are “outside” the eigenmodel $\Omega(X)$. Note that while $\nu^T \nu = I$, $\nu \nu^T \neq I$. Also, G is the projection of the $\Omega(Y)$ eigenspace onto $\Omega(X)$ (the U vectors), while Γ is the projection of $\Omega(Y)$ onto the complementary space to $\Omega(X)$ (the ν vectors). This complementary space must be determined to compute the new eigenspace $\Omega(Z)$, which argues in favour of adding and subtracting eigenspaces, rather than direct updating or downdating of data blocks.

Given the above decomposition, we can complete our computation of $\Omega(Z)$:

$$\Lambda(Z)_s = \text{diag}(\Pi_{ss}) \quad (16)$$

$$U_{ns}(Z) = [U_{np} \nu_{nt}] R_{ss} \quad (17)$$

The eigenmodel can then be deflated, if desired, to dimension $r \leq s$.

Each matrix in the above eigendecomposition is of size $s = p + t \leq p + q + 1 \leq \min(n, M + N)$. Thus we have eliminated the need for the original covariance matrices. Note this also reduces the size of the central matrix on the left hand side of Equation 8. This is of crucial computational importance because it makes the eigenproblem tractable for problems in which n is very large, such as when each datum is an image.

2.2 Subtraction

The algorithm for subtraction is very similar to that for addition. First compute the number of data, and their mean:

$$N(X) = N(Z) - N(Y) \quad (18)$$

$$\mu(X) = (N(Z)\mu(Z) - N(Y)\mu(Y))/N(X) \quad (19)$$

In this case $U(Z)$ is a sufficient spanning set to rotate. To compute the rotation we use the eigendecomposition:

$$\frac{N(Z)}{N(X)}\Lambda(Z)_{rr} - \frac{N(Y)}{N(X)}G_{rp}\Lambda(Y)_{pp}G_{rp}^T - \frac{N(Y)}{N(Z)}g_r g_r^T = R_{rr}\Lambda(X)_{rr}R_{rr}^T \quad (20)$$

where $G_{rp} = U(Z)_{nr}^T U(X)_{nq}$ and $g_r = U(Z)_{nr}^T (\mu Y - \mu X)$. The eigenvalues we seek are the p non-zero elements on the diagonal of $\Lambda(X)_{rr}$. Thus we can permute R_{rr} and $\Lambda(X)_{rr}$, and write without loss of generality:

$$\begin{aligned} R_{rr}\Lambda(X)_{rr}R_{rr}^T &= [R_{rp}R_{rt}] \begin{bmatrix} \Lambda(X)_{pp} & 0_{pt} \\ 0_{tp} & 0_{tt} \end{bmatrix} [R_{rp}R_{rt}]^T \\ &= R_{rp}\Lambda(X)_{pp}R_{rp}^T \end{aligned} \quad (21)$$

where $p = r - q$. Hence we need only identify the eigenvectors in R_{rr} with non-zero eigenvalues, and compute the $U(X)_{np}$ as:

$$U(X)_{np} = U(Z)_{nr}R_{rp} \quad (22)$$

Splitting must always involve the solution an eigenproblem of size r .

2.3 Some comments on the solutions

Previously we presented a method for adding a single point, x , to an eigenspace [10]. It can be shown [9] that the addition of exactly one new datum is a special case of the above addition, with $(x, 0, 0, 1)$ as either operand. In terms of its outcome the above addition is both commutative and associative (provided that in practice we allow for numerical errors, especially in association). The null eigenspace $(0, 0, 0, 0)$ is an additive identity. The addition of an eigenspace to itself yields an eigenspace which is identical in all respects except the number of points (which doubles). As $N(X) \rightarrow \infty$ so the effect of adding $\Omega(Y)$ becomes negligible, and vice-versa. As both $N(X)$ and $N(Y)$ tend to infinity together, so the result tends to a stable state.

The time complexity for addition will shadow that used in computing the eigenvalue decomposition. Our experiments [9] demonstrate that the time taken is $O(s^3)$, where s is the size of the eigenproblem to be solved (we used a proprietary eigensolver). We also found that the time to compute two eigenspaces *ab initio* and add them is about that of computing a large eigenspace using all the original data. However, it is much more efficient to

add a pair of existing eigenspaces than to compute their sum *ab initio*. Similar remarks apply to splitting: removing a few data points is a comparatively efficient operation. The conclusion we reach is that addition and subtraction of eigenspaces is no less efficient than batch methods, and in most cases is performed much more efficiently.

We have compared the angular deviation of eigenvectors, change in eigenvalues, accuracy of data representation in the least-squares sense, and classification performance [9]. The incremental methods for addition generally compare very well with batch methods, with discrepancies being a minimum when the two eigenspaces added are of about the same size; the exception is the discrepancy in eigenvalues, which shows a maximum of about one part in 10^5 at that point. Reasons for this behaviour are the subject of future work — we have not yet undertaken a rigorous analysis of errors.

The subtraction operator tends to instability as the number of points being removed rises, since in this case $N(X) \rightarrow 0$, hence $1/N(X) \rightarrow \infty$. In the limit of all points being removed $N(X) = 0$, and an exception must be coded to return a null eigenspace. Unfortunately, we have found that prior scaling by $N(X)$ to be ineffective and have concluded that, in practice, subtraction is best used to remove a small fraction of the data points.

3 Applications

An obvious application of our methods is to build an eigenspace from many images — too many to all at once fit into memory. We ran a simulation of this by building two eigenmodels: one using batch methods and another using our incremental methods. We were then able to compare the two models. The eigenspaces themselves turn out to be very similar, although differences between batch and incremental eigenspaces are greater in cases where eigenspaces are subtracted. Performance results bear out intuition: those images used to make the eigenspace had a much lower residue error than those not so used. As more images were added into the construction the maximum residue error for each image rose — but never so high as to reach the minimum residue error for images not used in eigenspace construction. Classification results follow a similar trend: each image is better classified by an eigenspace that uses all images.

We now present two more substantial applications of our methods. These are of a generic nature. The intention is to furnish the reader with a practically useful appreciation of the characteristics of our methods, and avoid the specific problems of any particular application.

3.1 Building an accurate eigenspace model

Here we consider an image database application. The scenario is that of a university wishing to efficiently store photographs of its thousands of students for use in a security application of some kind, such as access to a laboratory. The students are to be identified from their facial appearance. This problem is well researched, and we do not claim to make a contribution, rather we aim to show how our methods might be used in a support role. In particular, we consider the case in which the database of images changes, as old students leave and new ones arrive.

We proceed in a very simple way: we construct an eigenmodel of all those people who are to be recognised, and rely on the fact that eigenmodels do not generalise well to distinguish between those people in the set, and those not in the set. To allow for changes in pose, expression, and so on, we use several images of each individual.

Conventional batch methods cannot be used to construct an eigenmodel because there are too many images to fit into memory at once, so incremental methods are a pre-requisite to our approach. Given that the database is subject to change we could reconstruct an eigenmodel at each change, but we will use our incremental methods to effect the changes more efficiently; for which subtraction is required.

We used the Olivetti database of 400 faces ¹ as our group of students. We constructed an eigenmodel from a selection of 20 people, there being 10 photographs for each person. Each person in the entire database was then given a “weight of evidence” between 0 (not in the database) and 1 (in the database). To compute the weight we computed the maximum Mahalanobis distance (using Moghaddam and Pentland’s method [12]) of any photograph used in constructing the database. Each photograph was then classified as “in” if its Mahalanobis distance was less than this. Since each person has 10 photographs associated with them, we can then compute a weight for each person as the fraction of their photographs classified as in.

Figure 2 shows the “weight of evidence” measure for the *second* year our hypothesised database has been running. The scenario is that in year one, only persons 0 to 21 inclusive were “in”, while in year two only persons 1 to 22 inclusive were “in”. The leftmost plot shows the measure for the images against a batch model. That on the right shows the same measure for the same images, but for a model incrementally computed from year one by first including any new students (person 22), and then removing old students (person 0). (The ordering used to make sure the fraction of images removed was minimised.)

¹ <http://www.cam-orl.co.uk/facedatabase.html>

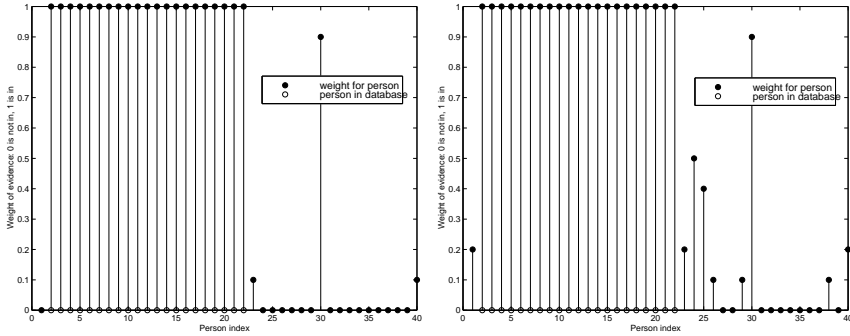


Fig. 2. Weight of evidence measures: year 2 batch (left), and year 2 incremental (right).

We notice that both models produce some ambiguous cases, with weights between 0 and 1, and that the incrementally computed eigenspace gives rise to more of these cases than the eigenmodel computed via batch methods. This result is in line with our earlier comments regarding the relative inaccuracy of subtraction. Even so, only those people “in” the database score 1, while everyone outside scored less than 1 and hence classification is still possible.

Given our observations, above, regarding previous measures when subtracting eigenspaces, we conclude that *additive* incremental eigenanalysis is safe for classification metrics, but that *subtractive* incremental eigenanalysis needs a greater degree of caution.

3.2 Dynamic Gaussian mixture models

We are interested in using our methods to construct dynamic GMMs. Gaussian mixture models are useful in computer vision contexts [5]. Our approach treats a GMM as a hierarchy of eigenspaces, which is a mechanism for improving the specificity of the data description [11]. To construct a hierarchy we first make an eigenmodel, then project all data into it to reduce dimensionality, next construct a GMM using the projected data, and then represent each mixture as an eigenmodel. Thus, each Gaussian in the mixture can be thought of as a hyperellipse, and each may have a different dimensions. The problem here is to merge two such GMMs.

As an example, we used photographs of two distinct toys, each photographed at 5 degree angles on a turntable. Hence we had 144 photographs. Examples of these photographs can be seen in Figure 3. The photographs for each toy were input separately, and a hierarchy of eigenmodels constructed as described above; we used eighteen Gaussians in each mixture model, on the grounds that this would very probably produce too many Gaussians — a number we would later improve by merging.

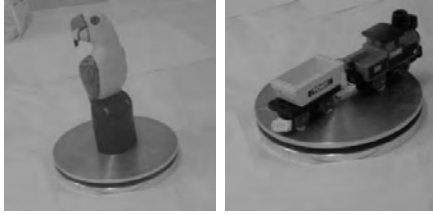


Fig. 3. Sample images of each toy used as source data in our dynamic GMM application.

Thus, including the top-level eigenspace, each set of toy photographs was represented with nineteen eigenspace models. To merge the GMMs for the pair of toys we first added together the two top-most eigenspaces to make a complete eigenspace for all 144 photographs. Next we transformed each of the GMM clusters into this space, thus bringing each of the thirty-six GMMs (eighteen from each individual hierarchy) into the same (large) eigenspace covering the *ensemble* of data. We then merged eigenspaces (Gaussian components), using a very simple criterion to merge based on reducing volume of hyperellipses, which is explained below. Hence, we were able to reduce the total number of Gaussians to 22 in the mixture. These clusters tend to model different parts of the cylindrical trajectories of the original data projected into the large eigenspace. Examples of cluster centres are shown in Figure 4: the two models can be clearly seen in different positions. In addition, we found a few clusters occupying the space “in between” the two toys — an example of which is seen in Figure 4.

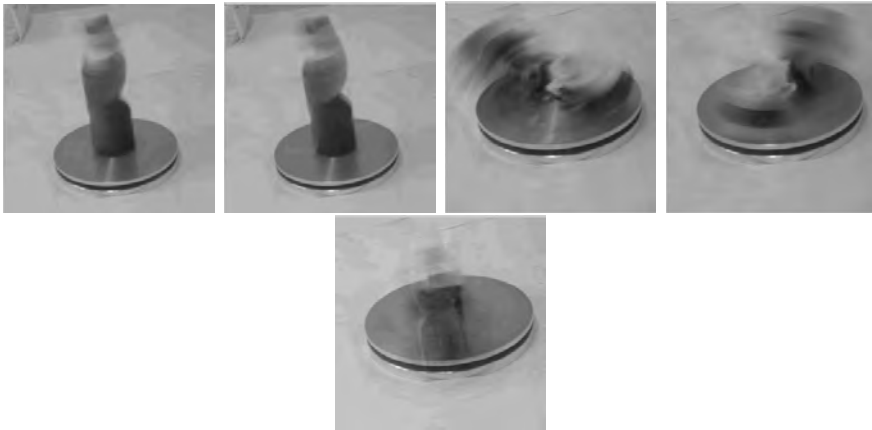


Fig. 4. Dynamic Gaussian Mixture Models, showing 5 examples of the 22 cluster centres. These are arranged to show clusters for each toy (top row), and the clusters between them (bottom).

As mentioned above, we used a simple method based on volume to decide whether two eigenspaces should be merged. The procedure was as follows. First compute the volume of each of the eigenmodels, using the hyperellipse at one Mahalanobis distance. The volume of a hyperellipse with semi-axes \mathbf{A} (each element the square root of an eigenvalue), of dimension M , and at characteristic radius s (square root of the Mahalanobis distance) is

$$\frac{s^M |\mathbf{A}| \pi^{M/2}}{\Gamma(\frac{M}{2} + 1)} \quad (23)$$

We permanently merged a pair of eigenmodels in the GMM if the sum of their individual volumes was greater than their volume when merged. This measure suffers from problems of dimension: we should not compare the volume of a p -dimensional hyperellipse with that of a q -dimensional hyperellipse. A solution is to use a characteristic length in place of volume, which for a p -dimensional hyperellipse of volume v is $v^{1/p}$.

Of course, the utility and properties of the final GMM is fully in line with any produced by conventional means, and hence can be used in any application that a conventional GMM is used. We conclude from these experiments that dynamic GMMs are a feasible proposition using our methods.

4 Comments on SVD approaches

This paper has focussed on block updating and block downdating of eigenspace models, based on eigenvalue decomposition (EVD). However EVD can suffer from conditioning problems (the condition number being the ratio of the smallest to largest eigenvalue). Singular value decomposition (SVD) tends to be more stable (and hence accurate) because singular values are proportional to the square root of eigenvalues, which mitigates conditioning problems. Thus we have also investigated block updating and block downdating based on singular value decomposition, which we briefly outline next.

We are given a set of N observations, each an n -dimensional column vector. (Note that sets here can contain repeated elements.) These observations can be represented by an $(n \times N)$ matrix, X . The mean of these observations is the vector μ . The SVD for the origin-centred data set is $X - \mu \mathbf{1} = (U \Sigma V^T) / \sqrt{(N)}$, where $\mathbf{1}$ is a row vector of N 1's. The left singular vectors are columns of U , and are identical to the eigenvectors in the EVD of XX^T . The right singular vectors V are related to the coordinates of the data when transformed into the U basis (which is the eigenspace): $V = ((X - \mu \mathbf{1})^T U \Sigma^{-1}) / \sqrt{(N)}$. The singular values, Σ , give the length of semi-axes along each U vector and specify the size of a hyperellipse at unit Mahalanobis distance. The singular values are related to the eigenvalues in the EVD of XX^T , $\Sigma \propto \Lambda^{1/2}$. As for eigenmodels, the system can be deflated by discarding left and right singular vectors that are associated with small singular values: $X - \mu \mathbf{1} \approx (U_{np} \Sigma_{pp} V_{Np}^T) / \sqrt{(N)}$. So, for data X we specify an SVD model as $\Phi(X) = (\mu(X), U(X), \Sigma(X), V(X), N(X))$.

The block updating problem for data sets X and Y is to compute the SVD for their union, conveniently written $Z = [X, Y]$, given only their SVD models

$\Phi(X)$ and $\Phi(Y)$. This is done in three stages. First the number $N(Z)$ and mean $\mu(Z)$ are computed as for EVD updates. An orthonormal basis set ν which spans any subspace of $U(Y)$ not in $U(X)$ is also computed in a way similar to that for EVD, but need not include the difference between the means — that is accounted for elsewhere in the SVD formulation. Second, the following singular value decomposition is made:

$$U \Sigma V^T = \begin{bmatrix} \sqrt{N(X)} \Sigma(X) V(X)^T & \sqrt{N(Y)} U(X)^T U(Y) \Sigma(Y) V(Y)^T \\ 0 & \sqrt{N(Y)} \nu U(Y) U(Y) \Sigma(Y) V(Y)^T \end{bmatrix} + \begin{bmatrix} U(X)^T (\mu(X) - \mu(Z)) \mathbf{1}_{N(X)} & U(X)^T (\mu(Y) - \mu(Z)) \mathbf{1}_{N(Y)} \\ \nu^T (\mu(X) - \mu(Z)) \mathbf{1}_{N(X)} & \nu^T (\mu(Y) - \mu(Z)) \mathbf{1}_{N(Y)} \end{bmatrix} \quad (24)$$

Here the second term accounts for the difference in means. Finally, the left and right singular vectors, and singular values are computed, and deflated as desired.

$$U(Z) = \nu U \quad (25)$$

$$S(Z) = S / \sqrt{N(Z)} \quad (26)$$

$$V(Z) = V \quad (27)$$

We note that the right singular values are given directly, a shift of mean is accounted for, and the problem is of size $(p+q) \times (N(X) + N(Y))$. Contrasting our solution with others [3], they add a single new point at time, do not shift the mean, and require post manipulation multiplication of right singular values without offering any efficiency gain in terms of problem size.

Downdate of SVD models means removing points in data set Z which are also in data set Y ; in set-theoretic terms we assume Y is a subset of Z and want to compute the SVD for $X = Z \setminus Y$. This is not straightforward and difficulties arise in two areas, even if we neglect a change in mean. The first difficulty comes from the (simplest) form of the problem which is $[ABC^T, DEF^T] = GHJ^T$, where $X = ABC^T$, $Y = DEF^T$, and $Z = GHJ^T$. We must obtain A , B , and C . By computing the inner product of both sides we obtain $AB^2A^T + DE^2D^T = GH^2G^T$, which gives us an EVD problem from which we can compute A and B . Alternatively, we can use the relationship between EVD and SVD to compute A , B , and the mean shift using the EVD downdating methods described above. However, we note that either way SVD downdate cannot be directly achieved.

The second difficulty arises when we note that the ordering of right singular vectors depends upon the ordering of data points in the matrix being decomposed. However, the left singular vectors and singular values are invariant to

permutation of the data. To see this we suppose P is a permutation matrix (obtained by randomly permuting the identity matrix, so that $PP^T = P^T P = I$), and note that given $Z = GHJ^T$, then $ZP = GHJ^T P = GH(P^T J)^T$. Therefore, in order to compute the right singular vectors C while downdating, we must have access to some matrix P which “picks out” data elements in Z (or, equivalently, corresponding elements in J). Unfortunately no such information exists within the SVD model, and consequently computing C in a closed-form manner seems impossible. The only solution seems to be a resort to search using data elements in J and F (for these specify data points in Z and Y respectively). If search is the only solution, then we may simply downdate Z by building up X incrementally — as elements in $Z \setminus Y$ are found — which is unsatisfactory in our opinion.

Thus we conclude that updating SVD models is possible, at the expense of keeping right singular values, and that downdate of SVD models is not possible in closed form. However, experimental evidence [10] suggests that SVD updates are likely to be more accurate than EVD updates.

5 Conclusion

We have presented methods for adding and subtracting eigenspaces. We have discussed the form of our solutions, and shown that previous work is a special case of this work. Our contribution is to track the mean in a principled way, which makes our contribution novel. This is essential in classification applications, which makes our contribution important.

Having experimentally compared eigenspaces, considered performance metrics of our algorithms, and having experimented with several more applications we have concluded that the addition of eigenspaces is stable and reliable. We advise that our methods be used carefully — any statistical method may be misapplied. Especial care should be taken when subtracting eigenspaces: the way in which the results are to be used impacts on efficacy.

We should point out several omissions from this work. We have not performed any rigorous error analysis and hence any explanations we have for the behaviour of our algorithms (in terms of approximating the “batch” version) are anecdotal in character. We have not fully worked through any particular application, and so can make general recommendations only. We have mentioned SVD techniques, and argued that SVD downdate is not possible in closed form, but have not experimentally compared block update of SVD with block update of EVD. We have conducted experiments which compare EVD and SVD update when adding a single new data point, and found SVD to be more accurate; see our earlier work [10]. This is an expected result, given that EVD

squares the condition number. We have also omitted comparisons with other incremental methods, because they deal with adding one new data point (but, again see our earlier work [10]). The important conclusion from that work is that updating the mean is crucial for classification results [10].

We would expect our methods to find much wider applicability than those we have already mentioned in this paper: updating image motion parameters [4], and selecting salient views [3] are two applications that exist already for incremental methods. We have experimented with image segmentation, building models of three-dimensional blood vessels, and texture classification. We believe that dynamic Gaussian mixture models provide a very interesting future path for they enable useful representations [5,11] — and all their attendant properties — to be brought into a dynamic framework.

References

- [1] James R. Bunch and Christopher P. Nielsen. Updating the singular value decomposition. *Numerische Mathematik*, 31:111–129, 1978.
- [2] James R. Bunch, Christopher P. Nielsen, and Danny C. Sorenson. Rank-one modification of the symmetric eigenproblem. *Numerische Mathematik*, 31:31–48, 1978.
- [3] S. Chandrasekaran, B.S. Manjunath, Y.F. Wang, J. Winkler, and H.Zhang. An eigenspace update algorithm for image analysis. *Graphical Models and Image Processing*, 59(5):321–332, September 1997.
- [4] S. Chaudhuri, S. Sharma, and S. Chatterjee. Recursive estimation of motion parameters. *Computer Vision and Image Understanding*, 64(3):434–442, November 1996.
- [5] T.F. Cootes and C.J. Taylor. A mixture model for representing shape variations. In *Proc. British Machine Vision Conference*, pages 110–119, 1997.
- [6] T.F. Cootes, C.J. Taylor, D.H. Cooper, and J. Graham. Training models of shape from sets of examples. In *Proc. British Machine Vision Conference*, pages 9–18, 1992.
- [7] Ronald D. DeGroat and Richard Roberts. Efficient, numerically stabilized rank-one eigenstructure updating. *IEEE Transactions on acoustics, speech, and signal processing*, 38(2):301–316, February 1990.
- [8] Gene H. Golub and Charles F. Van Loan. *Matrix computations*. Johns Hopkins, 1983.
- [9] Peter Hall, David Marshall, and Ralph Martin. Merging and splitting eigenspaces. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. (to appear)

- [10] Peter Hall, David Marshall, and Ralph Martin. Incrementally computing eigenspace models. In *Proc. British Machine Vision Conference*, pages 286–295, Southampton, 1998.
- [11] Tony Heap and David Hogg. Improving specificity in pdms using a heierarchical approach. In *Pooc. British Machine Conference*, pages 80–89, 1997.
- [12] Baback Moghaddam and Alex Pentland. Probabilistic visual learning for object representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(7):696–710, July 1997.
- [13] H. Murakami and B.V.K.V Kumar. Efficient calculation of primary images from a set of images. *IEEE Pattern Analysis and Machine Intelligence*, 4(5):511–515, September 1982.