Large-rank Approximate Principal Component Analysis with Wavelets for Signal Feature Discrimination and the Inversion of Complicated Maps

Mladen Victor Wickerhauser*
Department of Mathematics
Washington University in St. Louis
Missouri 63130 USA

June 6, 1994

Abstract

Principal component analysis (PCA) can be used to solve two related problems: distinguishing elements from a collection by making d measurements, and inverting a complicated map from a p-parameter configuration space to a d-dimensional measurement space. In the case where d is more than 1000 or so, the classical $O(d^3)$ singular value decomposition algorithm becomes very costly, but it can be replaced with an approximate best-basis method that has complexity $O(d^2 \log d)$. This can be used to compute an approximate Jacobian for a complicated map from $\mathbf{R}^p \to \mathbf{R}^d$ in the case $p \ll d$. This paper is a condensed version of an invited lecture given at *Math-Chem-Comp 1993* in Rovinj, Croatia, 21–25 June 1993.

1 Introduction

Consider the problem of most efficiently distinguishing elements from a collection by making d measurements. For example, the elements might be pictures of human faces and the measurements might be the recorded light intensities at discrete locations. Or the elements might be sampled NMR or IR spectra with the measurements being absorbances at discrete frequencies. In general, we will need all d measured values to fully specify an element. However, it is possible to use less information if some of the measurements are correlated. For example, if the objects are parameterized by a small number $p \ll d$ of parameters, then the d measurements should separate them in a redundant fashion. That is, we can change basis locally in the d-dimensional measurement space to find just p combinations of measurements which change with the p parameters. This idea works even if there are many parameters but only p of them are relatively important.

*Supported in part by AFOSR award F49620-92-J-0106 and NSF grant DMS-9302828.

Note the resemblance between the problem of distinguishing elements and the problem of inverting a complicated map from $\mathbf{R}^p$ to $\mathbf{R}^d$. In the first problem, we must find a discrete object given its description in $\mathbf{R}^d$. In the second, we must find the parameters in $\mathbf{R}^p$ from the description in $\mathbf{R}^d$. These problems are identical if the collection of objects is produced by evaluating the complicated map at discrete grid points in $\mathbf{R}^p$.

The combinations of measurements which root out the underlying parameters are called *principal (orthogonal) components* or *factors*; they have a precise meaning, and the well-known and well-behaved method of *singular value decomposition* [4] or *SVD* produces them with arbitrary accuracy. However, SVD has a computational complexity that is asymptotically $O(d^3)$, making it impractical for problems larger than $d \approx 1000$. In this paper, we will describe the classical principal component analysis (PCA) algorithm, then give a lower-complexity *approximate principal component analysis algorithm*. Finally, we will discuss applications of the approximate algorithm to the problems of distinguishing faces and fingerprints, and inverting the map which produces the IR spectrum of the atmosphere from the concentrations of absorbing gasses.

2 Notation and definitions

We need some definitions from probability theory.

2.1 Random vectors, means and variances

Let $X = \{X_n : n = 1, \ldots, N\} \subset \mathbf{R}^d$ be an ensemble of vectors. Write

$$E(X) \stackrel{\text{def}}{=} \frac{1}{N} \sum_{n=1}^{N} X_n \tag{1}$$

for the average vector in the ensemble, i.e., the expectation of x over the set X.

Let $\sigma(X) \subset \mathbf{R}^d$ be the vector of the standard deviations of the coefficients of X. Namely,

$$\sigma(X) = \sqrt{E(X^2) - E(X)^2} \tag{2}$$

$$\sigma(X)(k) = \left(\frac{1}{N} \sum_{n=1}^{N} [X_n(k) - E(X)(k)]^2 \right)^{1/2} \tag{3}$$

We define the *variance ellipsoid* of an ensemble X to be the ellipsoid centered at $E(X) \in \mathbf{R}^d$, with semiaxes $\sigma(X)(1), \sigma(X)(2), \ldots, \sigma(X)(d)$ aligned with the d coordinate axes. Its volume is $\omega_d \times [\sigma(X)(1)] \times [\sigma(X)(2)] \times \cdots \times [\sigma(X)(d)]$, where ω_d is the surface area of the unit sphere in $\mathbf{R}^d$.

If the ensemble X is fixed forever, then we can assume without loss of generality that $\frac{1}{N} \sum_{n=1}^{N} X_n(k) = 0$ for all $k = 1, 2, \ldots, d$, namely that $E(X) = 0$, because this can always be arranged by subtracting the average vector $E(X)$ from each of $X_1, X_2, \ldots, X_N$. It results in a simpler formula for $\sigma(X)$:

$$E(X) = 0 \Rightarrow \sigma(X)(k) = \left(\frac{1}{N} \sum_{n=1}^{N} X_n(k)^2 \right)^{\frac{1}{2}} \tag{4}$$

Then the variance ellipsoid is centered at 0.

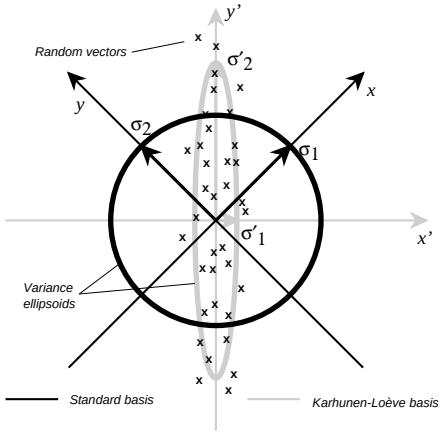


Figure 1: The variance ellipsoids for the standard and Karhunen–Loève bases

We write $\mathrm{Var}(X)$ for the total variance of the ensemble X. This is the sum of the squares of the coordinates in the variance vector $\sigma(X) \in \mathbf{R}^d$: $\mathrm{Var}(X) = \|\sigma(X)\|^2 \stackrel{\mathrm{def}}{=} \sum_{k=1}^{d} \sigma(X)(k)^2$, or

$$\mathrm{Var}(X) = \sum_{k=1}^{d} \left[\frac{1}{N} \sum_{n=1}^{N} X_n(k)^2 - \left(\frac{1}{N} \sum_{n=1}^{N} X_n(k) \right)^2 \right]. \tag{5}$$

2.2 The Karhunen–Loève transform

The *autocovariance matrix* for the ensemble X is defined by

$$M \stackrel{\mathrm{def}}{=} E(\bar{X} \otimes \bar{X}); \qquad M(i,j) = \frac{1}{N} \sum_{n=1}^{N} \bar{X}_n(i) \bar{X}_n(j). \tag{6}$$

Here we have taken $\bar{X}_n \stackrel{\mathrm{def}}{=} X_n - E(X)$ to be the original vector with the average vector subtracted. Thus $E(\bar{X}) = 0$. The matrix coefficient $M(i,j)$ is the covariance of the i^{th} and j^{th} coordinate of the random vector $\bar{X}$, using the ensemble as the sample space. The matrix M is evidently symmetric. It is also positive (some would say positive semidefinite) since for every vector $Y \in \mathbf{R}^d$ we have

$$\begin{aligned} \langle Y, MY \rangle &= \sum_{i=1}^{d} \sum_{j=1}^{d} Y(i) M(i,j) Y(j) \\ &= \frac{1}{N} \sum_{n=1}^{N} \sum_{i=1}^{d} \sum_{j=1}^{d} Y(i) \bar{X}_n(i) \bar{X}_n(j) Y(j) \\ &= \frac{1}{N} \sum_{n=1}^{N} \langle Y, \bar{X}_n \rangle^2 \geq 0 \end{aligned}$$

Therefore, we can find an orthonormal basis for $\mathbf{R}^d$ consisting of eigenvectors of the matrix M, and these eigenvectors will have positive real eigenvalues. Thus we are assured that the *Karhunen–Loève* basis exists for the ensemble X of vectors; this is the orthonormal basis of eigenvectors of the autocovariance matrix $E(\bar{X} \otimes \bar{X})$, as shown in Figure 1.

The Karhunen–Loève basis eigenvectors are also called *principal orthogonal components* or *principal factors*, and computing them for a given ensemble X is also called *factor analysis*. Since the autocovariance matrix for the Karhunen–Loève eigenvectors is diagonal, it follows that the coordinates of the vectors in the sample space X with respect to the Karhunen–Loève basis are uncorrelated random variables. Let us denote these basis eigenvectors by $\{Y_n : n = 1, \ldots, N\}$, and let us denote by K the $d \times d$ matrix whose columns are the vectors $Y_1, \ldots, Y_N$. The adjoint of K, or K^*, is the matrix which changes from the standard coordinates into Karhunen–Loève coordinates; this map is called the *Karhunen–Loève transform*.

Unfortunately, finding these eigenvectors requires diagonalizing a matrix of order d, which has complexity $O(d^3)$. In addition, even after already computing the Karhunen–Loève eigenvectors of an ensemble, updating the basis with some extra random vectors will cost an additional $O(d^3)$ operations since it requires another diagonalization.

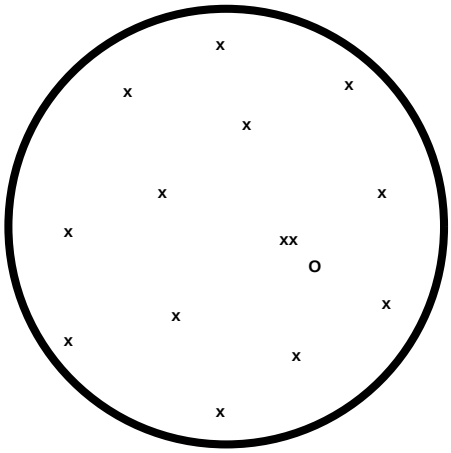
Such a high order of complexity imposes a ceiling on the size of problem we can do by this method. In many cases of interest, d is very large and X spans $\mathbf{R}^d$, implying $N \geq d$. Thus even to find the coefficients of the autocovariance matrix requires $O(d^3)$ operations. At the present time, we are limited to $d \leq 10^3$ if we must use common desktop computing equipment, and $d \leq 10^4$ for the very most powerful computers.

So we shall take another perspective. We shall pose the problem of finding the Karhunen–Loève eigenvectors as an optimization over the set of orthogonal transformations of the original ensemble X. The quantity to be maximized will be a *transform coding gain*, or the amount of compression we achieve by using another basis to represent the ensemble. Alternatively, we could simply introduce a distance measure on the set of orthonormal bases for $\mathbf{R}^d$, treat the Karhunen–Loève basis as a distinguished point in this set and then try to find an efficiently-computable basis which is close to the Karhunen–Loève basis.

2.3 A metric on the orthogonal matrices

Consider Figure 2, which schematically depicts all the orthonormal bases of $\mathbf{R}^d$. These can be identified with orthogonal transformations of $\mathbf{R}^d$. The points marked by "x" represent bases in some library of fast transforms. The point marked "o" represents the optimal, or Karhunen–Loève basis, for a given ensemble of vectors. The point marked "xx" represents the fast transform closest to the Karhunen–Loève bases.

Let U be an orthogonal $d \times d$ matrix, and write $Y = UX$ to signify that $Y_n = UX_n$ for each $n = 1, 2, \ldots, N$. Since U is linear, $E(Y) = E(UX) = UE(X)$, which will be 0 if we started with $E(X) = 0$. Since U is orthogonal, it preserves sums of squares, so $\mathrm{Var}(X) = \mathrm{Var}(Y)$. Using wavelet packets [1] or adapted local trigonometric functions [3], it is possible to build a library of more than 2^d fast transforms U of $\mathbf{R}^d$ to use for the "x" points. We will illustrate the construction using the simple Haar–Walsh wavelet packets in the example below. These transforms are arranged in a structure that permits us to search for the one closest to the "o" point in $O(d \log d)$ operations. We will use a notion of closeness that is derived from the function minimized by the Karhunen–Loève transform.



x = fast transform bases; o = Karhunen–Loève basis; xx = best fast basis.

Figure 2: Orthonormal bases for $\mathbf{R}^d$

As in [5], define the *transform coding gain* for an orthogonal matrix by the formula

$$G_{TC}(U) = \mathrm{Var}(UX)/\exp H(UX), \quad \text{where} \quad H(X) = \sum_{i=1}^{d} \log \sigma(X)(i). \tag{7}$$

From this formula we see that $G_{TC}(UX)$ is maximized when $H(UX)$ is minimized. The quantity H has various interpretations. It is the entropy of the direct sum of d independent Gaussian random variables with variances $\sigma(X)(i)$, $i = 1, \ldots, d$. It is also equal to the logarithm of the volume of the variance ellipsoid (if we add $\log \omega_d$), so we see that minimizing $H(UX)$ or maximizing $G_{TC}(UX)$ is equivalent to minimizing the volume of the variance ellipsoid for the ensemble UX over all orthogonal matrices U.

The Karhunen–Loève transform is therefore a global minimum for H, and we will say that the best approximation to the Karhunen–Loève transform from a library $\mathcal{U}$ of orthogonal matrices is the minimum of $H(UX)$ with the constraint $U \in \mathcal{U}$. We can define the *approximate factor analysis algorithm* to be the searches through a library of orthonormal bases for the one closest to the Karhunen–Loève basis. If the library of bases is organized to facilitate a fast search, we will dare to call the result a *fast approximate Karhunen–Loève algorithm.*

The "closeness" of a basis U to the Karhunen–Loève basis K can be measured by computing the transform coding gain of U and subtracting that of K. This give us a *transform coding gain metric*:

$$\delta_X(U, V) = |H(UX) - H(VX)|$$

Notice that we get a different metric for each ensemble X. This is a degenerate metric on the orthogonal group, since it gives a distance of 0 between bases which have the same transform coding gain for X. However, this technical point can be overcome by constructing a topological quotient space in which such bases are considered equivalent.

The Karhunen–Loève basis $V = K$ is a minimum for $H(VX)$, so a minimum of $H(UX)$

over fast transforms U also minimizes $\delta_X(U, K)$ and finds the fast transform closest to the Karhunen–Loève transform for this ensemble in the sense of efficiency of representation.

3 The approximate KL transform

We will now describe a large library of rapidly-computable orthonormal bases, constructed by a recursive decomposition algorithm which takes advantage of the rapid growth of the number of subtrees of a binary tree. We also describe an efficient search algorithm which can be used to maximize transform coding gain (or minimize entropy) over all the bases which fall from the tree. The vectors which make up these bases are discrete approximations to *wavelet packets*, which are described in detail in several papers [1, 3, 2]. We will fix our attention on the *Haar–Walsh wavelet packets*. It is relatively easy to generalize this example to the other wavelet packet approximate Karhunen–Loève expansions.

Let S and D be two operators which, together with their respective adjoints S^* and D^*, are defined on sequences by the following formulas:

$$Sx(n) = [x(2n) + x(2n+1)]/\sqrt{2}; \tag{8}$$

$$S^*x(n) = \begin{cases} \frac{1}{\sqrt{2}}x(\frac{n}{2}), & \text{if } n \text{ is even,} \\ \frac{1}{\sqrt{2}}x(\frac{n-1}{2}), & \text{if } n \text{ is odd;} \end{cases} \tag{9}$$

$$Dx(n) = [x(2n) - x(2n+1)]/\sqrt{2}; \tag{10}$$

$$D^*x(n) = \begin{cases} \frac{1}{\sqrt{2}}x(\frac{n}{2}), & \text{if } n \text{ is even,} \\ -\frac{1}{\sqrt{2}}x(\frac{n-1}{2}), & \text{if } n \text{ is odd.} \end{cases} \tag{11}$$

S is the so-called *low-pass filter* and D is the *high-pass filter* conjugate quadrature filter. They are also called *Haar-Walsh filters*, since they are used to produce Haar and Walsh bases. The factors $1/\sqrt{2}$ insure that the operators conserve energy and variance. S^* and D^* are the adjoint filters, and it is easy to see that $SS^* = DD^* = I$, $SD^* = DS^* = 0$ and $S^*S + D^*D = I$. Those facts are used to establish some remarkable orthogonality relations among wavelet packets.

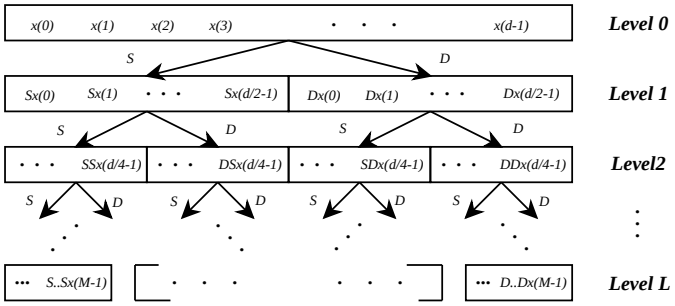


Figure 3: Decomposition into Haar–Walsh wavelet packets

Starting with a signal $x = \{x(n) : n = 0, 1, \ldots, d-1\}$ of $d = M2^L$ samples, we can recursively apply S and D a total of L times because the number of samples is divisible by 2 at least L times. We arrange the resulting sequences in a binary tree as in Figure 3. Notice that if we take

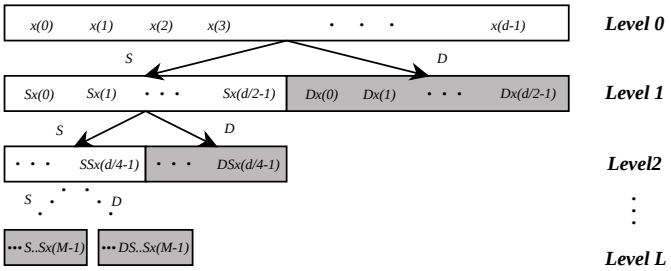


Figure 4: Example graph basis: the Haar wavelets

a "graph" of blocks from that tree, namely a collection with the property that each vertical line goes through exactly one block, then the numbers in the blocks are coefficients with respect to an orthonormal basis. Each graph gives a different orthonormal basis; for example, the shaded blocks depicted in Figure 4 contain the so-called "wavelet basis" coefficients.

We then sum the coefficients of each tree into two "accumulator" trees:

- the tree of *means*, which contains $\sum_{n=0}^{N-1} DSSD\ldots Dx_n(k)$ in location k of the block labeled $DSSD\ldots D$, and so on;
- the tree of *squares*, which contains $\sum_{n=0}^{N-1} [SDSD\ldots Dx_n(k)]^2$ in location k of the block labeled $SDSD\ldots D$, and so on.

Computing all the coefficients of all the blocks in an L-level tree starting from d samples takes $O(dL) = O(d\log d)$ operations per random vector, for a total of $O(Nd\log d)$ operations. After we do this for all the random vectors X_n in the ensemble X, we can produce the binary tree of *variances* by using Equation 2: at index k of block $DSD\ldots S$, for example, it contains

$$\sigma^2_{DS\ldots D}(X)(k) \stackrel{\text{def}}{=} \frac{1}{N}\sum_{n=0}^{N-1} DS\ldots Dx_n(k)^2 - \left[\frac{1}{N}\sum_{n=0}^{N-1} DS\ldots Dx_n(k)\right]^2$$

This is the variance of the wavelet packet coefficient defined by the filter sequence $DSD\ldots S$ and the location k. Forming this tree takes an additional $O(d\log d)$ operations. For convenience we introduce the following notation for these blocks which compactly encodes filter sequences such as $DSDSSD$. Let $n = (n_{L-1}\ldots n_1 n_0)_2$ be the binary representation of the nonnegative integer $0 \le n < 2^L$. Then $n = n_{L-1}2^{L-1} + \cdots + n_1 2^1 + n_0$ with $n_i \in \{0,1\}$ for $i = 0, 1, \ldots, L-1$. This will represent the filter sequence $F_0F_1\ldots F_{L-1}$, where

$$F_i = \begin{cases} S, & \text{if } n_i = 0, \\ D, & \text{if } n_i = 1. \end{cases}$$

Block n at level j of the variance tree will be denoted V_{jn}; its coefficients form the sequence which may be denoted

$$(\sigma_{jn}(X)(0), \sigma_{jn}(X)(1), \ldots, \sigma_{jn}(X)(d/2^j - 1)).$$

Notice that the two children of block V_{jn} are the S block $V_{j+1,2n}$ and the D block $V_{j+1,2n+1}$, both at level $j+1$.

The tree of variances may now be searched for the graph basis which minimizes H. We begin by defining the *information cost function*:

$$\mathcal{H}(V_{jn}) \stackrel{\text{def}}{=} \sum_{k=0}^{d/2^j-1} \log \sigma_{jn}(X)(k). \tag{12}$$

Since this calculation will always be done at some fixed finite precision $\epsilon > 0$, we never really have a singularity and we will simply replace $\log 0$ by $\log \epsilon$.

Now let I_{jn} be the information cost of the best graph basis in the subtree whose root is V_{jn}. Let U_{jn} be the collection of blocks in that best graph basis. In other words, $\sum_{V \in U_{jn}} \mathcal{H}(V)$ is minimal with all the blocks $V = V_{j'n'}$ in the graph U_{jn} being descendants of V_{jn} and satisfying $j' \geq j$. Then we can find the minimum for the whole tree by the following preorder recursion:

BestGraphBasis(j, n, L, V, I_{jn}, U_{jn})**:**
Find the best graph basis and its least information cost.

Inputs:

j, the current level,

n, the current block index,

L, the maximum depth,

V, the tree of all wavelet packet subspaces $V_{j'n'}$. We need at least those subspaces "below" V_{jn}, namely with $j' = j, j+1, \ldots, L$ and $n' = 2^{j'-j}n, \ldots, 2^{j'-j}(n+1) - 1$.

Outputs:

I_{jn}, the best information cost of any graph basis for V_{jn},

U_{jn}, the best graph basis for V_{jn}.

Implementation:

- Compute $I_{jn} = \mathcal{H}(V_{jn})$ using Equation 12.
- If $j = L$, then we are at a bottom-level subspace, so
  - Retain I_{jn} as the information cost.
  - Set $U_{Ln} = \{V_{Ln}\}$,
- Else $j < L$, so we compare V_{jn} against the lower levels:
  - Find $U_{j+1,2n}$ and $I_{j+1,2n}$ recursively with BestGraphBasis($j+1$, $2n$, L, V, $I_{j+1,2n}$, $U_{j+1,2n}$).
  - Find $U_{j+1,2n+1}$ and $I_{j+1,2n+1}$ recursively with BestGraphBasis($j+1$, $2n+1$, L, V, $I_{j+1,2n+1}$, $U_{j+1,2n+1}$).
  - If $I_{jn} \leq I_{j+1,2n} + I_{j+1,2n+1}$ then V_{jn} by itself gives as good a representation of X as $U_{j+1,2n} \cup U_{j+1,2n+1}$, so
    - Retain I_{jn} as the information cost;
    - Set $U_{jn} = \{V_{jn}\}$, a single-block graph.
  - Else $I_{jn} > I_{j+1,2n} + I_{j+1,2n+1}$, so it is cheaper to represent X using the descendant blocks in $U_{j+1,2n} \cup U_{j+1,2n+1}$, so
    - Set $I_{jn} = I_{j+1,2n} + I_{j+1,2n+1}$;

* Set $U_{jn} = U_{j+1,2n} \cup U_{j+1,2n+1}$, the union of the best graph bases for the descendants of V_{jn}.

- Return.

This recursion terminates leaving the best graph basis in U_{00} and the lowest information cost of any graph basis in I_{00}. A formal proof of these facts can be found in [3].

Notice that each block is examined twice: once when it is a parent and once when it is a child. This means that the best-basis search requires as many comparison operations as there are blocks in the tree, which is $O(d)$. Computing $\mathcal{H}$ costs no more than a fixed number of arithmetic operations per coefficient in the tree, which is $O(d \log d)$. Thus the total cost of the search is $O(d \log d)$.

U_{00} will be called the *joint best basis* for the ensemble X, in the Haar–Walsh wavelet packet library. We can denote by U the $d \times d$ orthogonal matrix which corresponds to the orthonormal basis U_{00}. Abusing notation, we write $\{U_i \in \mathbf{R}^d : i = 1, \ldots, d\}$ for the rows of U. We may suppose that these rows are numbered so that $\sigma(UX)$ is in decreasing order; this can be done by sorting all the d coefficients in all the blocks $V \in U_{00}$ into decreasing order, which can be done in $O(d \log d)$ operations.

If we fix $\epsilon > 0$ and let d' be the smallest integer such that

$$\sum_{n=1}^{d'} \sigma(UX)(n)^2 \geq (1 - \epsilon)\mathrm{Var}(X),$$

then the projection of X onto the row span of $U' = \{U_1 \ldots U_{d'}\}$ contains $1-\epsilon$ of the total variance of the ensemble X. Call this projected ensemble $\tilde{X}$. The d' row vectors of U' are already a good basis for the ensemble $\tilde{X}$, but they may be further decorrelated by Karhunen–Loève factor analysis. The row vectors of U' are just $U'_i = U_i$ for $1 \leq i \leq d'$, and the autocovariance matrix for this new collection is given by

$$M'_{ij} = \frac{1}{N} \sum_{n=1}^{N} U'_i \bar{X}_n U'_j \bar{X}_n. \tag{13}$$

Here $\bar{X}_n = \tilde{X}_n - E(\tilde{X})$ is a vector in $\mathbf{R}^d$, $U'\bar{X}$ is a vector in $\mathbf{R}^{d'}$, and $E(U'\bar{X}) = 0$. Thus M' is a $d' \times d'$ matrix and can be diagonalized in $O(d'^3)$ operations. Let K' be the matrix of singular vectors of M'. Then K'^* changes from the joint best basis coordinates (calculated from the standard coordinates by U') into coordinates with respect to these decorrelated singular vectors. We may thus call the composition K'^*U' (associated to ϵ) the *approximate Karhunen–Loève transform with relative variance error* ϵ.

The algorithm is fast because we expect that even for small ϵ we will obtain $d' \ll d$. The resulting relatively low complexity of the approximate Karhunen–Loève transformation makes it useful for very large rank problems where traditional principal factor analysis would be too costly. We shall now count the operations needed to perform each step, to understand where we save effort and to characterize when the method can be expected to work.

To pose the general case, which has no degeneracy or artificially low rank, we assume the following:

1. N random vectors;
2. d-dimensional parameter space $\mathbf{R}^d$;
3. Full-rank autocovariance matrix ($N > d$).

These assumptions allow us to count operations as we implement each step of the algorithm. The first step is to compute the joint best basis, or the fast transform basis which is closest to Karhunen–Loève:

JointBestBasis(X, N, d, Σ_1, Σ_2, U, S)

Find the fast transform closest to Karhunen–Loève for an ensemble of vectors

Inputs:

X, an $N \times d$ array or N vectors of length d;

N, the number of vectors in X;

d, the dimension of each vector in X.

Outputs:

Σ_1, the sums of the wavelet packet coefficients of the ensemble X, organized into a tree;

Σ_2, the sums of the squares of the wavelet packet coefficients of the ensemble X, organized into a tree;

U, a graph basis for d-dimensional signals which is the approximate Karhunen–Loève basis for the ensemble X;

S, an ordered array of pointers into U which can be used to list basis elements in decreasing order of variance.

Implementation:

- Expand the N vectors $\{X_n \in \mathbf{R}^d : n = 1, 2, \ldots, N\}$ into (temporary) wavelet packet coefficient trees $V^{(1)}, \ldots V^{(N)}$, using Equations 8,10 and Figure 3. This costs $O(Nd \log d)$ operations;
- Sum the coefficients in $V^{(1)}, \ldots, V^{(N)}$ into the means tree Σ_1. This costs $O(Nd \log d)$ operations;
- Sum the squares of the coefficients in $V^{(1)}, \ldots, V^{(N)}$ into the square tree Σ_2. This costs $O(Nd \log d)$ operations;
- Compute the variances of the coefficients in $V^{(1)}, \ldots, V^{(N)}$ from Σ_1 and Σ_2, and superpose the values into the (temporary) variance tree Var using Equation 3. This costs $O(d \log d)$ operations;
- Search the variance tree for a best basis using the function BestGraphBasis(0,0,Var,I,U), where I is a temporary variable used to hold information costs. This takes $O(d + d \log d)$ operations;
- Sort the coefficients in the best basis U of Var into decreasing order using a low average complexity sorting algorithm like "quick sort" or "heap sort", and list them in decreasing order in an array S. This takes $O(d \log d)$ operations.

Adding up all the operations gives a complexity of $O(Nd \log d) = O(d^2 \log d)$.

Next, we find the matrix which decorrelates the top few joint best basis components of an ensemble of vectors:

ApproximateKLMatrix(X, N, d, ϵ, U, S, K'^*, d')

Decorrelate the vectors which capture all but ϵ of the total variance

Inputs:

X, an $N \times d$ array or N vectors of length d;

N, the number of vectors in X;

d, the dimension of each vector in X.

ϵ, the fraction of the total variance we are allowed to discard.

U, the joint best graph basis for the ensemble X;

S, the array of pointers into U listing basis elements in decreasing order of variance;

Outputs:

K'^*, a $d' \times d'$ matrix which decorrelates the first d' most varying elements in the approximate Karhunen–Loève basis;

d', the number of approximate Karhunen–Loève vectors we need to capture $1 - \epsilon$ of the total variance of the ensemble X.

Implementation:

- Find d' by summing up variances in U in the order given by S until we have $1 - \epsilon$ of the total variance. This takes $O(d')$ operations;
- Compute the coefficients in the basis U of the N vectors $\{X_n \in \mathbf{R}^d : n = 1, 2, \ldots, N\}$ and copy them into (temporary) vectors $X'_1, \ldots, X'_N$ of length d' each, using Equations 8,10 and Figure 3. This costs $O(Nd \log d)$ operations. We can trade space for time if we keep the trees $V^{(1)}, \ldots, V^{(N)}$ from JointBestBasis() and simply extract the U-basis sets. Then the complexity is $O(Nd')$, but the storage space is $O(Nd \log d)$;
- Find the autocovariance matrix M of the ensemble $\{X'_1, \ldots, X'_N\}$, using Equation 6. It takes $O(Nd'^2)$ operations.
- Diagonalize the $d' \times d'$ autocovariance matrix M using singular value decomposition, and make the singular vectors into the rows of K'^*. This costs $O(d'^3)$ operations.

This part of the algorithm requires a total of $O(Nd \log d + d'^3)$ operations, or $O(Nd' + d'^3)$ if we retain the trees from JointBestBasis().

Adding these up gives the total complexity of constructing the approximate Karhunen–Loève transform K'^*U', which is $O(Nd \log d + d'^3)$. This compares favorably with the complexity $O(Nd^2 + d^3)$ of the full Karhunen–Loève expansion, since we expect $d' \ll d$.

Depending on circumstances, the last step $U' \mapsto K'^*U'$ may not be necessary if a large reduction in the number of parameters is already achieved by transforming into the orthonormal basis determined by U'. This reduces the complexity to $O(Nd \log d)$, with the penalty being less decorrelation of the factors.

ApproximateKLTransform(v, d, U, S, K'^*, d', w)
The approximate Karhunen–Loève transform of one vector

Inputs:

v, a d-component input vector;

d, the dimension of the vector v.

U, the approximate Karhunen–Loève graph basis we use for d-dimensional signals;

S, an ordered array of pointers into U which listing basis elements in decreasing order of variance;

K'^*, a $d' \times d'$ matrix which decorrelates the first d' most varying elements in the approximate Karhunen–Loève basis;

d', the number of approximate Karhunen–Loève vectors we need to capture $1 - \epsilon$ of the total variance of the ensemble X.

Output:

w, a d'-element array which will hold the coefficient of the input vector with respect to the decorrelated approximate Karhunen–Loève basis.

Implementation:

- Compute the complete wavelet packet expansion of v and put the coefficients into the (temporary) tree V. This takes $O(d \log d)$ operations;
- Form the temporary vector z by extracting the top d' coefficients of v in the U-basis subset of V, using the ordering defined by S. This costs $O(d')$ operations.
- Compute $w = K'^* z$, applying the $d' \times d'$ matrix K'^* which is defined after Equation 13 to the d'-component vector z. This costs $O(d'^2)$ operations.

Adding these up gives $O(d \log d + d'^2)$ for the total complexity of computing one approximate Karhunen–Loève transform. When $d' \ll d$, this estimate compares favorably with the complexity of applying the full Karhunen–Loève transform to a vector, which is $O(d^2)$. Further savings are possible, notably because only a small fraction $d'' \ll d'$ of the Karhunen–Loève singular vectors are needed to capture almost all of the original ensemble variance. Hence we can take K'' to be just the first d'' of the columns of K', and then the total complexity of applying $K''^* U'$ is bounded by $O(d \log d + d'' d')$.

If we expect to update the joint best basis, then we might also expect to update the average vector and the average value of each coordinate in the library of bases, as well as the variance of the ensemble. But since we keep a sum-of-squares tree and a sum-of-coefficients or means tree rather than a variance tree, each additional random vector just contributes its wavelet packet coordinates into the means tree and the squares of its coordinates into the sum-of-squares tree. The variance tree is updated at the end using the correct new means. This is done by JointBestBasis(Y, m, d, Σ_1, Σ_2, U, S), where Y is a new ensemble of m vectors and Σ_1,Σ_2 are the means and squares trees carried over from the previous calculations. The resulting update complexity is reduced in proportion to the small number of new vectors. One new vector costs $O(d \log d)$, and updating the joint best basis with $m > 1$ new vectors costs $O(md \log d)$.

4 Classification in large data sets

The Karhunen–Loève transform can be used to reparameterize a problem so as to extract prominent features with the fewest measurements. When the number of measurements is huge, the fast approximate algorithm must be used at least as a "front end" to reduce the complexity of the SVD portion of finding the Karhunen–Loève basis.

We list a few examples to give some indication of the size of problem that can be treated by the approximate method on typical tabletop computing equipment.

4.1 The rogues' gallery problem

The "rogues' gallery" problem is to identify a face from among a collection of faces. This probelm was first suggested to me by Lawrence Sirovich, who also provided the data used in this experiment. The random vectors were several thousand digitized 128×128 pixel pictures of Brown University students, so $d = 128^2 = 16,384$. The pixel intensities are integers in the range $\{0, 1, \ldots, 255\}$. The pictures were manually scaled and translated so that each student's eyes came to rest on two fixed points near the center of the image. In [7, 6], a supercomputer was used to compute the Karhunen–Loève transform either of the complete set of pixels or else of an oval subset centered about the eyes. In the following, we will follow Sirovich's methodology and nomenclature, only we will replace the Karhunen–Loève transform with the lower-complexity approximate algorithm.

Figure 5: Face, minus the average face, yields a caricature.

For the experiment described below, we start with a more limited data set containing 143 pictures. Since the ensemble was fixed, we could subtract the average vector at the outset. Thus we transformed the data to floating point numbers, computed average values for the pixels, and then subtracted the average from each pixel to obtain "caricatures," or deviations from the average. Figure 5 is one of these caricatures:

The left graph in Figure 6 shows how the variance accumulates pixel by pixel, with the pixels sorted into decreasing order of variance.

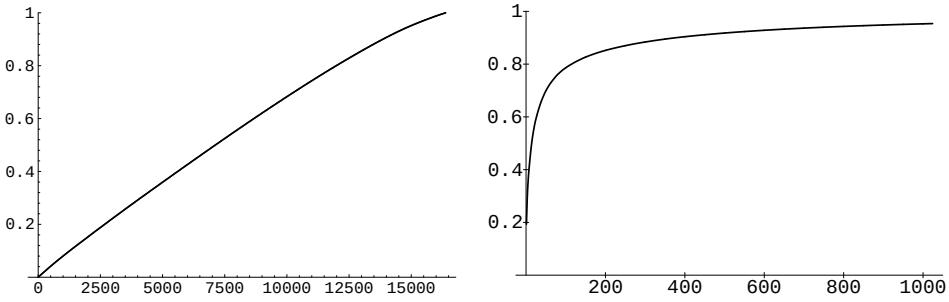


Figure 6: Accumulation of variance in the original basis and the joint best basis.

Each caricature was treated as a picture and expanded into 2 dimensional wavelet packets

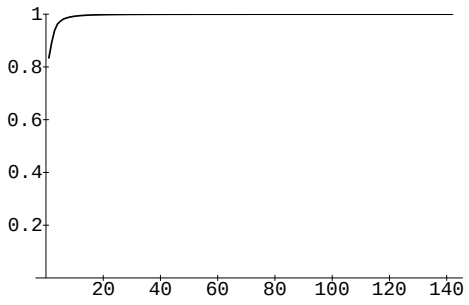


Figure 7: Accumulation of variance in the approximate Karhunen–Loève basis.

as described in [8]. The squares of the amplitudes were summed into a tree of variances, which was then searched for the joint best basis. In the joint best basis, 400 coordinates (of 16,384) contained more than 90% of the variance of the ensemble.

The right graph in Figure 6 shows the accumulation of total variance on the first d' coordinates in the joint best basis, sorted in decreasing order, as a fraction of the total variance of the ensemble, for $1 \leq d' \leq 1000$. Using 1000 parameters captures more than 95% of the ensemble variance, but requires somewhat more computer power than is readily available on a desktop. A 400 parameter system, on the other hand, can be analyzed on a typical workstation in minutes so we chose $d' = 400$.

The top 400 coordinates were recomputed for each caricature and their autocovariance matrix over the ensemble was diagonalized using the LINPACK singular value decomposition routine.

Figure 7 shows the accumulation of total variance on the first d'' coordinates in the approximate Karhunen–Loève basis, sorted in decreasing order, as a fraction of the total variance of the 400 joint best basis coefficients, for $1 \leq d'' \leq 143$. The Karhunen–Loève post-processing for this small ensemble concentrates 98% of the retained variance from the top 400 joint best-basis parameters into 10 coefficients.

4.2 The fingerprint classification problem

Virtually the same method as the one applied to faces can be applied to fingerprints for identification purposes. The United States FBI uses 8 bits per pixel to define the shade of gray and stores 500 pixels per inch, which works out to about 600,000 pixels and 0.6 megabytes per finger to store fingerprints in electronic form. This means that $d \approx 10^6$, so we are forced to use the fast approximate algorithm if we wish to compute the Karhunen–Loève expansion of a fingerprint.

There is no apparent relation between the parameters chosen by the Karhunen–Loève transform and the traditional parameters (location of "minutiae" points in a fingerprint) which are used by police inspectors to describe fingerprints. Thus the additional classifying values would have to be stored alongside the more traditional values. No one is likely to complain, though, since the Karhunen–Loève parameters occupy only a few hundred bytes, which is a negligible amount of space compared to the million bytes of raw data.

4.3 Rank reduction for complex classifiers

The principle behind the fast approximate Karhunen–Loève transform is to employ a relatively low-complexity $O(d^2 \log d)$ "front end" to reduce the rank of the subsequent high-complexity $O(d^3)$ algorithm from d to $d' \ll d$.

If some measurements on an ensemble of random vectors are to be processed further by some other complex algorithm, we may similarly gain a significant speed advantage by pre-processing the data to reduce the number of parameters. A typical example would be processing for statistical classification from a large set of measurements. Classes, or regions in the measurement space $\mathbf{R}^d$, may have complicated boundaries which must be approximated by high-order polynomial hypersurfaces. Deciding at high orders whether a point lies in a particular region becomes very expensive when the dimension d grows large, so a reduction in the number of parameters will gain speed even if does not by itself simplify the geometry of the regions.

High complexity classifiers are used in speech recognition and machine vision systems. Adding a front end to reduce their workload is like adding a hearing aid or glasses so they can better focus on the most-varying features. In some cases, the speed is desirable because we wish to perform the classification in "real time" or least fast enough to keep up with the inflowing data. Some examples are:

- Mechanical failure detection from strain gauge data: $d \approx 10^2$;
- Target recognition from 1-dimensional radar range profiles: $d \approx 10^2$;
- Detection of irregular heartbeats from acoustic samples: $d \approx 10^2$;
- Phoneme detection: $d \approx 10^3$;
- Optical character recognition: $d \approx 10^3$;
- Detection of machine tool breakage from acoustic samples: $d \approx 10^3$;

5 Jacobians of complicated maps

Suppose that $T : \mathbf{R}^p \to \mathbf{R}^d$ is some smooth vector field with $p \ll d$. We may think of making plenty (d) of measurements of Tx for a variable x with only a few (p) degrees of freedom. This situation models one course of action to determine what a complicated map T is doing.

5.1 Approximating the tangent space in principal components

Recall that the *Jacobian* of T at a point $x \in \mathbf{R}^p$ is the $d \times p$ matrix $J = J_T[x]$ which gives the best linear approximation to T in a neighborhood of x. The coefficients of J are the various partial derivatives of T.

$$J_T[x](i,j) \stackrel{\text{def}}{=} \lim_{r \to 0} \left\langle e_i, \frac{T(x + re_j) - T(x)}{r} \right\rangle, \tag{14}$$

Here $1 \le i \le d$, $1 \le j \le p$, and e_i is the i^{th} standard basis vector. However, the numerical computation of this Jacobian poses some difficulties because the difference quotient is ill-conditioned. Furthermore, the Jacobian might itself be an ill-conditioned matrix, but this difference quotient procedure offers no way of estimating the condition number of J. We will address these difficulties by replacing the difference quotient formula with an approximation based on the

Karhunen–Loève transform for the positive matrix JJ^*. The error will lie solely in the approximation, since the Karhunen–Loève transform is orthogonal and thus perfectly conditioned. We will estimate the condition number of J from the singular value decomposition of J^*J. Then

$$cond(J) = \sqrt{cond(J^*J)} \approx \sqrt{\mu_1/\mu_p}, \tag{15}$$

where μ_1 and μ_p are respectively the first and n^{th} singular values of our estimate for J^*J.

Suppose first that T is a linear map, so that $T = J$ is its own Jacobian. Fix $x \in \mathbf{R}^p$, and consider the ball $B_r = B_r(x) \stackrel{\text{def}}{=} \{y \in \mathbf{R}^p : \|y - x\| \leq r\}$ of radius $r > 0$, centered at x. We can consider the image $JB_r = \{Jy : y \in B_r(x)\} \subset \mathbf{R}^d$ of this ball under the transformation J to be an ensemble of random vectors. This will have expectation $E(JB_r) = JE(B_r) = Jx$, and we can compute the autocovariance matrix of the zero-mean ensemble $\overline{JB_r} \stackrel{\text{def}}{=} JB_r - Jx$ as follows:

$$\begin{aligned} E(\overline{JB_r} \otimes \overline{JB_r}) &= E_{y \in B_r(x)}(J\bar{y}\,[J\bar{y}]^*) \\ &= JE_{y \in B_r(x)}(\bar{y}\bar{y}^*)J^* = r^2 JJ^*. \end{aligned}$$

Here $\bar{y} = y - x$ and the last equality holds since $E_{y \in B_r(x)}(\bar{y}\bar{y}^*) = r^2 I_d$ is just a constant times the $d \times d$ identity matrix and thus commutes out from between J and J^*. Thus $r^{-2}E(\overline{JB_r} \otimes \overline{JB_r}) = JJ^*$.

Proposition 1 *For every matrix J,*

$$Rank\ JJ^* = Rank\ J = Rank\ J^* = Rank\ J^*J.$$

Proof: To prove the first equality, notice that Range $JJ^* \subset$ Range J implies that Rank $J \geq$ Rank JJ^*. Now suppose that Range $JJ^* \neq$ Range J. There is some $y \neq 0$ with $y \in$ Range J and $\langle y, JJ^*z\rangle = \langle J^*y, J^*z\rangle = 0$ for all z. Putting $z = y$ we see that $J^*y = 0$. But by its definition, $y = Jx$ for some x, so we have $\|y\|^2 = \langle y, Jx\rangle = \langle J^*y, x\rangle = 0$, a contradiction. The third equality follows from the same argument if we substitute J^* for J. For the middle equality, note that Rank $AB \leq \min\{\text{Rank } A, \text{Rank } B\}$, so that the first equality gives Rank $J \leq$ Rank J^* while the third gives Rank $J \geq$ Rank J^*. □

Suppose J is a $d \times p$ matrix with $d \geq p$. If J has maximal rank p, then J^*J also has rank p. Now, the condition number of J is

$$\left(\sup_{x \neq 0} \frac{\|Jx\|}{\|x\|}\right)\left(\inf_{y \neq 0} \frac{\|Jy\|}{\|y\|}\right)^{-1} \tag{16}$$

If J^*J has p nonzero singular values $\mu_1 \geq \cdots \geq \mu_p > 0$, counting multiplicities, then the supremum is $\sqrt{\mu_1}$ and the infimum is $\sqrt{\mu_p}$. To see this, let $z_1, \ldots, z_p$ be the orthonormal basis of $\mathbf{R}^p$ consisting of unit singular vectors for J^*J, which is guaranteed to exist because J^*J is a Hermitean matrix. Writing $x = \sum_i a_i z_i$ we have

$$\frac{\|Jx\|^2}{\|x\|^2} = \frac{\langle Jx, Jx\rangle}{\|x\|^2} = \frac{\langle J^*Jx, x\rangle}{\|x\|^2} = \frac{\sum_i a_i^2 \mu_i}{\sum_i a_i^2}. \tag{17}$$

This average is maximized when just a_1 is nonzero and minimized when just a_p is nonzero; it then equals μ_1 and μ_p, respectively. Thus we can compute the condition number of J using the formula in Equation 15.

Now suppose that T is any smooth vector field from $\mathbf{R}^p$ to $\mathbf{R}^d$, and x is some point in $\mathbf{R}^p$. We compute $z_r = E(TB_r)$, where $TB_r = TB_r(x) = \{Ty : \|y - x\| \leq r\}$; this is the expected value of Ty for y in the ball $B_r(x)$ of radius r centered at x. This average gives a second-order approximation to Tx:

Proposition 2 $\|z_r - Tx\| = O(r^2)$ *as* $r \to 0$.

Proof: We can use Taylor's theorem to write $T(x + y) = Tx + Jy + O(\|y\|^2)$ for $\|y\| \leq r$. But $E(Jy) = JE(y) = 0$ because we evaluate the expectation over $y \in B_r(0)$. Thus $E(TB_r) = Tx + O(r^2)$. □

We now define a positive matrix

$$A = A_r = E([TB_r - z_r] \otimes [TB_r - z_r]), \tag{18}$$

where the expectation is taken over the ball of radius r. Our main theorem is the following:

Theorem 3 $\lim_{r \to 0} \frac{1}{r^2} A_r = JJ^*$.

Proof: Using Proposition 2 we write $z_r = Tx + O(r^2)$. We then use Taylor's theorem to get the following estimate:

$$\begin{aligned} [T(x+y) - z] \otimes [T(x+y) - z] &= \left[Jy + O(\|y\|^2)\right] \otimes \left[Jy + O(\|y\|^2)\right] \\ &= (Jy)(Jy)^* + O(\|y\|^3). \end{aligned}$$

Taking the expectation of both sides over $y \in B_r(0)$ gives $A_r = r^2 JJ^* + O(r^3)$, yielding the desired result. Notice that we also get an error estimate: $\|JJ^* - \frac{1}{r^2} A_r\| = O(r)$. □

Now suppose that $x \in \mathbf{R}^p$ is a point for which the Jacobian $J_T[x]$ has full rank p. Full rank is an open condition, *i.e.*, is it possessed by all matrices sufficiently close to J as well, so we have the following reassuring fact:

Corollary 4 *For all sufficiently small* $r > 0$, *Rank* $A_r \geq$ *Rank* $J = p$. □

Then we can approximate the map T in a neighborhood of Tx using the singular vectors for A_r:

Corollary 5 *If* $\{z_1, \ldots, z_p\} \subset \mathbf{R}^d$ *is a set of unit orthogonal singular vectors for* A_r, *then there are* p *linear functions* $c_1, \ldots, c_p$ *on* $\mathbf{R}^p$ *such that* $T(x + y) = z + \sum_{i=1}^{p} c_i(y) z_i + O(r\|y\|^2)$. □

We are not really concerned with the rank of A_r being too small, since A_r is a $d \times d$ matrix and $d \gg p$ is the interesting situation. Rather, we worry that choosing a too-large value for r will result in Rank A_r being too large, so that we will not be able to identify the top few singular vectors which approximately span Range J. The problem is that if T has nonvanishing higher-order derivatives, then the range of A_r will be higher-dimensional than the tangent space to T at Tx, which is the range of J. Schematically, this is depicted in Figure 8. The range of A_r is drawn as the two unit singular vectors z_1, z_2 multiplied by their respective singular values μ_1, μ_2. Notice that $\mu_2 \ll \mu_1$, illustrating what happens with smooth T: the variation μ_2 of TB_r in the nontangential direction z_2 is much smaller than the variation μ_1 in the tangent or z_1 direction. In practice, we will always have Rank $A_r = d$ because to a finite precision machine every approximate matrix looks like it has full rank. However, if we arrange the singular values of A_r (with multiplicity) in decreasing order $\mu_1 \geq \cdots \mu_p \geq \cdots \geq 0$, then for small enough r we expect a steep drop between μ_p and μ_{p+1}. This then provides a method of choosing the largest r for which the singular vectors of A_r provide an accurate parametrization of T near x. Namely, we let r increase until $\sqrt{\mu_{p+1}/\mu_p}$ reaches some preset threshold of precision ϵ_μ.

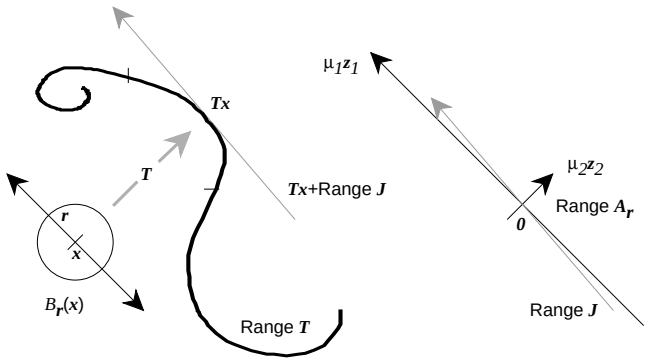


Figure 8: Tangent space (Range J) of T and its approximation (Range A_r)

Then the nontangential components will contribute an error which is $1/\epsilon_\mu$ times smaller than the tangential components.

The functions $c_1, \ldots, c_p$ in Corollary 5 correspond to partial derivatives, but they are computed by orthogonal projection. We define matrix coefficients c_{ij} using the elementary basis vectors e_j of $\mathbf{R}^p$ as follows:

$$C_{ij} \stackrel{\text{def}}{=} \frac{1}{r} \langle z_i, T(x + re_j) - z \rangle \tag{19}$$

Then we extend this to the superposition $y = \sum_{j=1}^{p} a_j e_j$ by taking linear combinations as follows:

$$c_i(y) \stackrel{\text{def}}{=} \sum_{j=1}^{p} C_{ij} a_j. \tag{20}$$

Comparing Equation 19 with Equation 14, we see that the $d \times p$ matrix J has been replaced with the $p \times p$ matrix C, the limit has been reduced to a single evaluation using the largest acceptable r, the initial point Tx is now an average z, and the standard basis $\{e_j : j = 1, \ldots, d\}$ in the range space has been replaced with a new orthonormal basis which is locally adapted to T.

Notice that the columns of the $p \times p$ matrix $C = (C_{ij})$ are given by the top p coordinates of the Karhunen–Loève transform of the secant vectors

$$\frac{1}{r}\left[T(x + re_1) - z\right], \frac{1}{r}\left[T(x + re_2) - z\right], \ldots, \frac{1}{r}\left[T(x + re_p) - z\right],$$

since the unit singular vectors $z_1, \ldots, z_p$ of A_r are the Karhunen–Loève eigenvectors for the ensemble TB_r. These secant vectors are approximations to the directional derivatives of T in the directions $e_1, e_2, \ldots, e_p$, and the Karhunen–Loève transform projects them onto the principal orthogonal components along Range T.

5.2 Fast approximate Jacobians

We can use the fast approximate Karhunen–Loève algorithm to compute the approximate Jacobian. Suppose that the domain of T includes a cube centered at the origin, namely

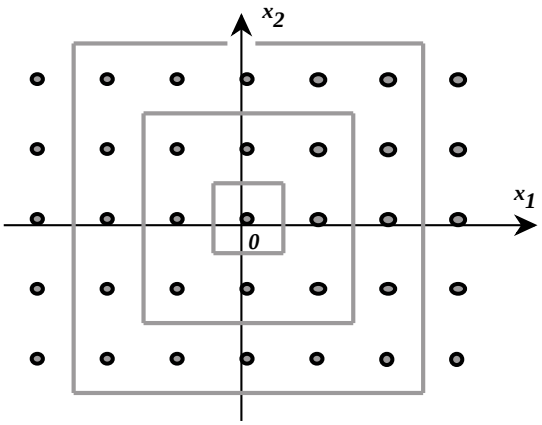


Figure 9: Cubes of various radii centered at 0

$B_r = B_r(0) \stackrel{\text{def}}{=} \{x \in \mathbf{R}^p : |x_1| \leq r, \ldots, |x_p| \leq r\}$. Suppose we lay down a uniform grid of points of the form $x_i = k$ where $k = 0, \pm 1, \pm 2, \ldots$ and $i = 1, \ldots, p$. The intersection of the cube with this grid, which we will also call B_r, contains $(2r+1)^p$ points in all. We now compute Tx at all points $x \in B_r$, and call the resulting set TB_r. This will be our ensemble of "random" vectors. Each $x \in B_r$ produces a vector $Tx \in \mathbf{R}^d$ which requires d numbers to store, so TB_r will contain $|B_r|d = (2r+1)^p \times d$ floating-point numbers.

The approximation to T at 0 using B_r will be computed by the wavelet packet best-basis algorithm above. The mean vector $z = E(TB_r)$ is computed in all wavelet packet bases at once in the means tree. We form the variance tree from the squares tree and the means tree, and search it for the joint best basis of TB_r. We may assume that this basis is sorted into decreasing order of variance.

Now we take d' of the most-varying terms out of the d in the joint best basis to retain $1-\epsilon$ of the total variance. We then form the $d' \times d'$ autocovariance matrix for these d' joint best basis vectors and diagonalize it, finding the singular vectors $z_1, \ldots, z_{d'}$ and corresponding singular values $\mu_1 \geq \ldots \geq \mu_{d'}$. We put the singular vectors into the columns of a matrix K', and get the *approximate Karhunen–Loève transform* K'^*, a $d' \times d'$ matrix which gives the approximate principal components when applied to the top d' joint best basis coordinates.

We now test whether the cube B_r is too large for the singular vectors to well-approximate the tangent space. The rank of the Jacobian is at most p, so we must have $\epsilon_\mu(r) = \sqrt{\mu_{p+1}/\mu_p} \ll 1$. This gives the first parameter of the algorithm: we know that $\epsilon_\mu(r) \to 0$ as $r \to 0$, so if it exceeds a preset threshold we need to reduce r. However, if $\epsilon_\mu(r)$ meets our requirements, then we can discard all but the first p columns of K' to get the $d' \times p$ matrix K''. The range of K'' serves as the approximate tangent space at $T0$, and K''^* computes coordinates in this space from the top d' joint best basis coordinates.

Finally, we form the approximate Jacobian into this approximate tangent space by using Equation 19 with an approximate principal factor for z_i. One by one, we take the secant vectors $\frac{1}{r}[T(x + re_j) - z]$ for $j = 1, 2, \ldots, p$, which live in $\mathbf{R}^d$, and we find their joint best basis expansions. We then extract from each of those expansions the previously-chosen d' coordinates

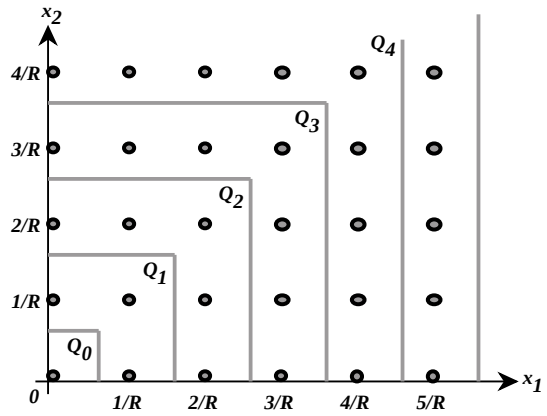


Figure 10: Patches of gridpoints at various radii from 0

which have most of the variance over the ensemble and apply K''^* to the vectors of these coordinates. That gives a list of p vectors in $\mathbf{R}^p$ which approximate the coefficients of the partial derivatives $\partial_1 T, \ldots, \partial_p T$ in the approximate tangent space basis.

The approximate Jacobian data consists of

- the joint best-basis description down to d' coordinates (d' numbers),
- the p vectors of the approximate tangent space expressed as combinations of the first d' joint best basis vectors (pd' numbers), and
- the $p \times p$ matrix C of partial derivatives expressed as combinations of the approximate tangent vectors.

Computing these quantities will cost us $O(|B_r| \times [d'^3 + d^2 \log d])$ arithmetic operations, where we expect $d' \ll d$.

5.3 Efficient storage of complicated maps

Suppose for this application that the domain of T is the unit cube in $Q \subset \mathbf{R}^p$ defined by $Q = \{x \in \mathbf{R}^p : 0 \leq x_1 \leq 1, \ldots, 0 \leq x_p \leq 1\}$, and suppose we lay down a uniform grid of points of the form $x_i = r/R$ where $r = 0, 1, \ldots, R$ and $i = 1, \ldots, p$. We will call this grid G; it has mesh $1/R$ and contains $|G| = (R+1)^p$ points in all. We now compute Tx at all points $x \in G$, and call the resulting set TG. This is an enormous data set: each $x \in G$ produces a vector $Tx \in \mathbf{R}^d$ which requires d numbers to store, so TG will contain $|G|d = (R+1)^p \times d$ floating-point numbers.

We now use approximate Jacobians to reduce the size of the data set. We will do this by building up patches in the domain where T is well-approximated by its approximate Jacobian. With the fast update algorithm, we can segment the domain of T into patches on which we are sure that the approximation remains within a preset error. Suppose we start at $0 \in G$, at one corner of the cube Q. For each $r = 0, 1, 2, \ldots$ we define the set $Q_r = Q_r(0) \stackrel{\text{def}}{=} \{x \in G : 0 \leq x_i \leq r/R \text{ for } i = 1, \ldots, p\}$. We also define the set $P_r = P_r(0) \stackrel{\text{def}}{=} Q_r(0) \setminus Q_{r-1}(0)$, the partial

cubical shell at radius r/R. Then $Q_{r+1} = Q_r \cup P_{r+1}$. This arrangement is depicted for the two-dimensional case $p = 2$ by the points enclosed in lightly-colored boxes in Figure 10. Note that Q_r contains $|Q_r| = (r+1)^p$ points, while P_r contains $|P_r| = |Q_r| - |Q_{r-1}| = (r+1)^p - r^p \approx pr^{p-1}$ points.

The segmentation algorithm works by first initializing $r = 1$, and then iterating through an algorithm that enlarges a patch on which T is linearly approximated by its approximate Jacobian. We stop enlarging the patch Q_r when the variance of TQ_r along the approximate tangent vectors to T stops being much larger than the variance along approximate normal vectors. In following, we assume that the approximate Karhunen–Loève basis for TQ_{r-1} has already been determined, using the algorithm in Section 3 above. The update algorithm is also defined in that section.

- Compute the wavelet packet means and sums-of-squares trees for the extra vectors TP_r;
- Update the joint best basis for TQ_{r-1} by adding in the data from TP_r to get the joint best basis for TQ_r;
- Compute and store the approximate Karhunen–Loève basis K'_r for TQ_r and the singular values $\mu_1 \geq \ldots \geq \mu_{d'} \geq 0$;
- If $\epsilon_\mu(r) = \sqrt{\mu_{p+1}/\mu_p}$ is too large, then:
  - Compute and store $z = ETQ_{r-1}$ for the center;
  - Compute and store the approximate tangent vectors K''_{r-1} for the patch TQ_{r-1};
  - Compute and store the approximate Jacobian using K''_{r-1} and Equation 19;
  - Reset $k = 1$;
  - Move to the next free point in G;
- Else if $\epsilon_\mu(r)$ is still small enough, then increment r by 1;
- Repeat.

This algorithm will eat away at the domain G, producing a covering of patches Q of various sizes, each with its center point $z_Q = ETQ$, its local approximate tangent space K''_Q and its approximate Jacobian C_Q. These quantities will require d', pd' and p^2 real numbers to store, respectively. If there are a total of N patches, then the total amount of data to store is $N(d' + pd' + p^2) = O(Npd')$ numbers since $p \leq d'$. If p is small, $N \ll |G|$ and $d' \ll d$, then this compares favorably with the storage requirements for TG, namely $O(d|G|)$.

The complexity of computing these quantities on all of the patches can be estimated from the complexity of finding the approximate Jacobian for the a single patch containing all of G, since this if the worst case. But from the previous section, we see that this is $O(|G| \times [d'^3 + d^2 \log d])$ arithmetic operations.

Applying this approximation of T to a vector $x \in \mathbf{R}^p$ involves first finding the patch Q with $x \in Q$. Suppose that x_Q is the center grid point of Q. Then in the first d' joint best-basis coordinates,

$$\widetilde{Tx} = z_Q + K''_Q C_Q (x - x_Q). \tag{21}$$

We finally superpose the d' joint best basis vectors to get the coordinates of the point $Tx \in \mathbf{R}^d$ from $\widetilde{Tx}$. The complexity of computing Tx this way is $O(p + p^2 + pd' + d' + d \log d)$ which under our assumptions is bounded by $O(d \log d)$.

5.4 Precomputation for inverse problems

The final application is to use the local approximate Jacobians to invert the map $x \mapsto Tx$, which we suppose has already been computed at all points x on a finite grid G.

One classical way is to use linear interpolation: given $y \in \mathbf{R}^d$, we find the nearest points $Tx_k \in \mathbf{R}^d$ computed from grid points $x_k \in G$ and write $y = \sum_k a_k Tx_k$. Then the linear approximation to $T^{-1}y$ is just $\sum_k a_k x_k$. This is exactly correct for linear maps T and has at least $O(h)$ accuracy for a differentiable map T on a grid with mesh h. However, it requires that we store the precomputed values $\{Tx : x \in G\}$ and that we search the whole list for the points close to y. The last step, in particular, requires computing $|G|$ distances for a grid G.

If we have invested the effort to compute the approximate Jacobian representation of T, then the inverse can be approximated from that data instead. Let N be the number of patches in the cover of G, and suppose that $N \ll |G|$. We also suppose for the sake of simplicity that we keep the same number d' of joint best basis components in each patch, although of course they may be different components in different patches. Finally, we suppose that we have already computed the inverses C_Q^{-1} of the approximate Jacobians on each patch Q, which requires a one-time investment of $O(Np^3)$. Then the necessary computations and their complexities for computing $T^{-1}y$ at a single $y \in \mathbf{R}^d$ are the following:

- Find the complete wavelet packet expansion of y, which simultaneously computes all joint best-basis expansions $\tilde{y}$: $O(d \log d)$;
- Compute the distances from $\tilde{y}$ to the means z_Q for each patch Q, and let Q henceforth be the patch with nearest mean: $O(Nd')$;
- Compute the approximate inverse,

$$T^{-1}y \approx x_Q + C_Q^{-1} K''^*_Q (\tilde{y} - z_Q) \tag{22}$$

  for the nearest-mean patch Q: $O(d' + pd' + p^2 + p) = O(pd')$.

References

[1] Ronald R. Coifman, Yves Meyer, Stephen R. Quake, and Mladen Victor Wickerhauser. Signal processing and compression with wavelet packets. In Yves Meyer and Sylvie Roques, editors, *Progress in Wavelet Analysis and Applications*, Proceedings of the International Conference "Wavelets and Applications," Toulouse, France, 8–13 June 1992, pages 77–93. Editions Frontieres, Gif-sur-Yvette, France, 1993.

[2] Ronald R. Coifman, Yves Meyer, and Mladen Victor Wickerhauser. Wavelet analysis and signal processing. In Mary Beth Ruskai, Gregory Beylkin, Ronald Coifman, Ingrid Daubechies, Stéphane Mallat, Yves Meyer, and Louise Raphael, editors, *Wavelets and Their Applications*, pages 153–178. Jones and Bartlett, Boston, 1992.

[3] Ronald R. Coifman and Mladen Victor Wickerhauser. Entropy based algorithms for best basis selection. *IEEE Transactions on Information Theory*, 32:712–718, March 1992.

[4] J. J. Dongarra, J. R. Bunch, C. B. Moler, and G. W. Stewart. *LINPACK User's Guide*. SIAM Press, Philadelphia, 1979.

[5] Nuggehally S. Jayant and Peter Noll. *Digital Coding of Waveforms: Principles and Applications to Speech and Video*. Prentice-Hall, Englewood Cliffs, New Jersey, 1984.

[6] Michael Kirby and Lawrence Sirovich. Application of the Karhunen–Loève procedure for the characterization of human faces. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12:103–108, January 1990.

[7] Lawrence Sirovich and Carole H. Sirovich. Low dimensional description of complicated phenomena. *Contemporary Mathematics*, 99:277–305, 1989.

[8] Mladen Victor Wickerhauser. High-resolution still picture compression. *Digital Signal Processing: a Review Journal*, 2(4):204–226, October 1992.