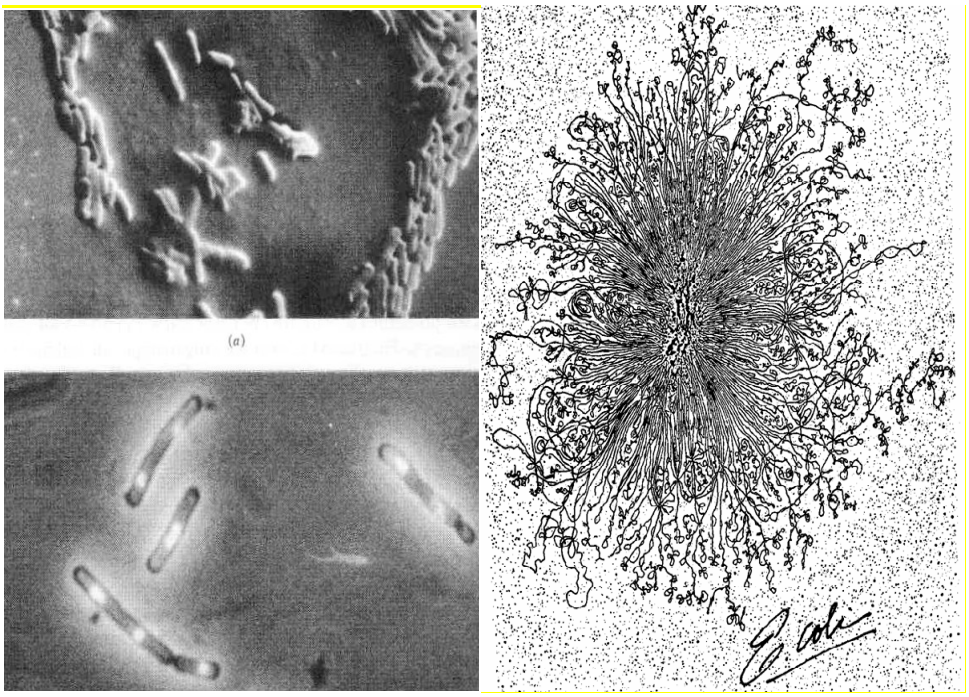
A. Assadi

Principal Component Analysis

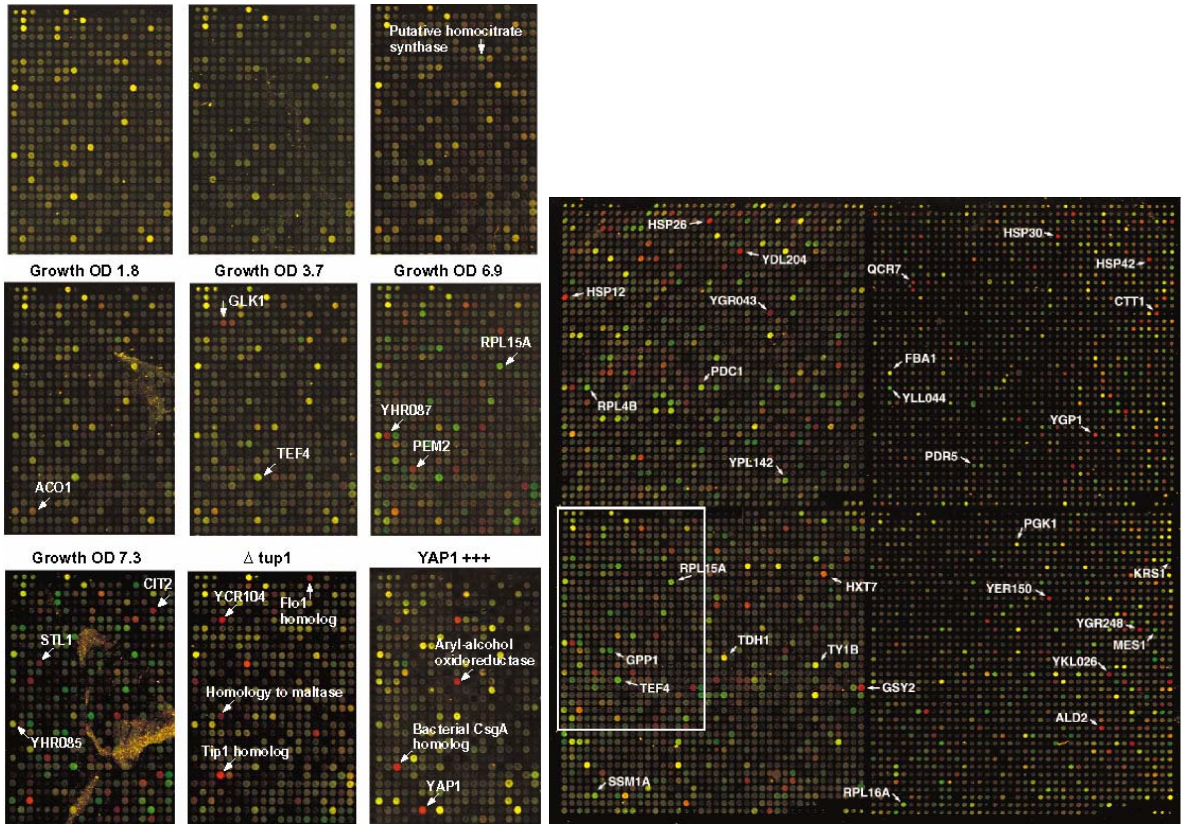
Introduction

One of the basic steps in analysis of data is to select a method for its representation that is best suited to the objectives of data analysis and organization of information. For instance, suppose that the data set that we call **D** consists of audio-visual signals that are digitally recorded, and the objective is to transfer this data across a network with a given band-width W in real-time, as in "video-on-demand" or other multi-media applications. As another application, suppose that the data set **D** consists of collections of sequences for genomes of strains of bacteria *E. coli,* and we wish to analyze their similarities and differences in structure, say in terms of nucleotides, loci of mutations, etc. Typically, such sequences have millions of bases and require a great deal of computer memory form storage of each one.



Electron Micrograph of a DNA Molecule Released from a Bacterium E. coli. This DNA is about one millimeter long or about 1000 times the length of the bacterium from which it was taken.

A third example is microarray data from gene chips, given as arrays of numbers that represent intensities of fluorescence recorded from gene chips. As a concrete case, consider the data collected from the budding yeas, as in DeRisi et. al. [Science, October 1997].

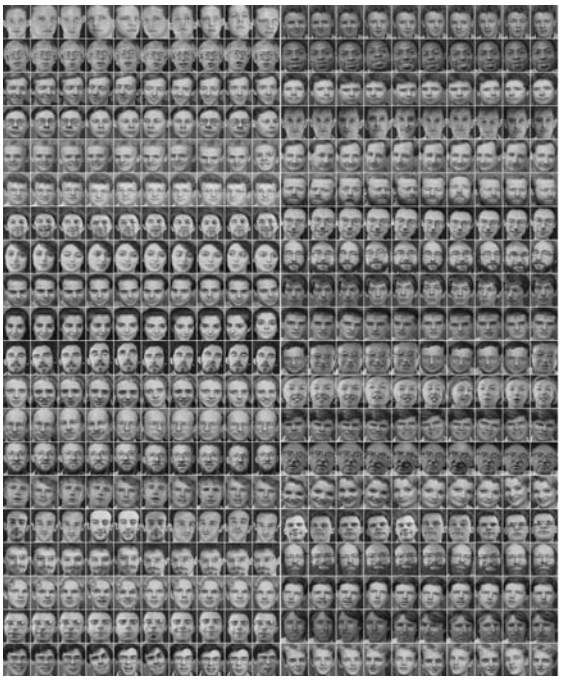


Yeast genome microarray. DNA microarrays containing virtually every gene of Saccharomyces cerevisiae ae used to carry out a comprehensive investigation of the temporal program of gene expression accompanying the metabolic shift from fermentation to respiration. The actual size of the microarray is 18 mm by 18 mm. This image was obtained with the fluorescent scanning confocal microscope. A fluorescently labeled cDNA probe was prepared from mRNA isolated from cells harvested shortly after inoculation by reverse transcription in the presence of Cy3-dUTP. Similarly, a second probe was prepared from mRNA isolated from cells taken from the same culture 9.5 hours later by reverse transcription in the presence of Cy5-dUTP. In this image, hybridization of the CyS-dUTP-labeled cDNA (that is, mRNA expression at the initial timepoint) is represented as a green signal, and hybridization of Cy5-dUTP-labeled cDNA (that is, mRNA expression at 9.5 hours) is represented as a red signal. Thus, genes induced or repressed after the diauxic shift appear in this image as red and green spots, respectively. Genes expressed at roughly equal levels before and after the diauxic shift appear in this image as yellow spots. In the Figure at left, the section of the array indicated by the gray box in Figure at right is shown for each of the experiments described here. Representative genes are labeled. In each of the arrays used to analyze gene expression during the diauxic shift, red spots represent genes that were induced relative to the initial timepoint, and green spots represent genes that were repressed relative to the initial time-point. In the arrays used to analyze the effects of the tup1/^ mutation and YAP1 overexpres-sion, red spots represent genes whose expression was increased, and green spots represent genes whose expression was decreased by the genetic modification. Note that distinct sets of genes are induced and repressed in the different experiments. Cell density as measured by optical density (OD) at 600 nm was used to measure the growth of the culture. (From DeRisi et. al. Science • Vol. 278 • 24 October 1997).

Let us use the notation T for of transformation of the data set **D**, and call the result of the data transformation T(**D**). Below, we mention some common requirements from T(**D**) that is often difficult to achieve in the original form of data.

Example 1. A typical example is where *compression of data* is an important by-product of the choice of data representation, as in the first example above. In real-time digital video transmission it is essential to reduce the size of the data that one wishes to transfer, because the total amount of data is very large compared to the visual and auditory information that is needed by the perceptual system for a realistic effect. In this case, the basic sets of parameters that play key roles for representation of data are: (1) a subjective measure for judging the quality of the transmitted audio-video signal T(**D**) compared to the quality of the play-back of the original

data; (2) the band-width W that determines the rate of transfer among other factors in using the network; (3) the size of the original data; (4) the computational resources available to us for the purpose of real-time data transfer across the network. In realistic applications, all four sets of parameters must be dealt with, and the modeling solutions must reflect a pragmatic compromise between the computational cost and the quality of the outcome. In simple terms, one must reduce the size of the transformed data T(**D**) as much as possible without losing too much of the visual and auditory quality. The general form of the real-time multi-media data transformation is often formulated in terms of digital representation of data by real numbers (primarily integers or rational numbers). In engineering and scientific applications, one often applies fast Fourier transform (FFT), wavelet transform, or their variants. It is usual to have signals that are naturally represented by arrays of complex numbers. The question arises as to how to generalize the preceding discussion to the case of complex vector spaces and complex vector representation of signals. More comments will be provided later on such generalizations. The following image consists of 40 rows, each having 10 poses from facial images of different people in jpeg format, greatly reduced in size.

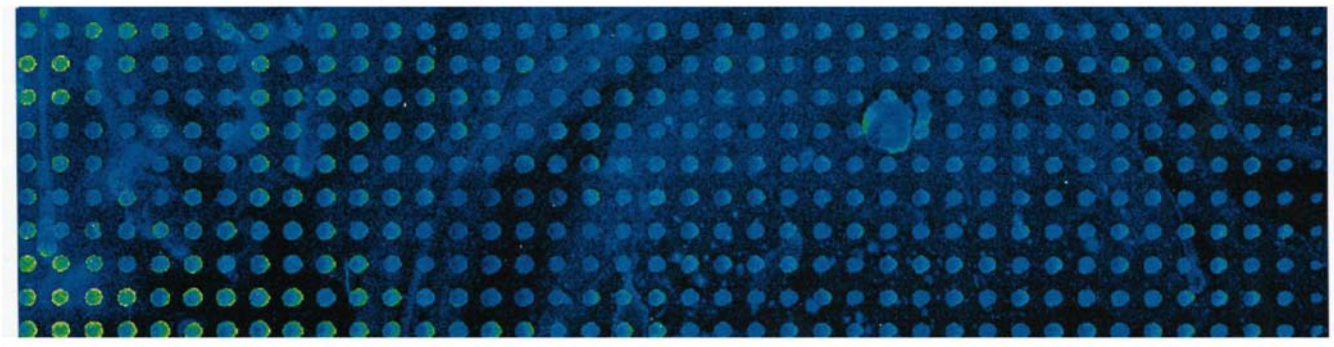


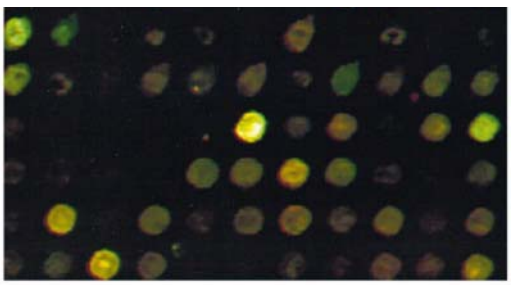
Each small individual photo is shown below using a 300 by 500 matrix of values between 0 and 255, indicating shades of gray given by 8 bits. The resolution of the image is 100 dots per inch in both dimensions. While the compressed size of the jpeg file is only 3.4 KB, the storage size of such single image on disk is 16,384 bytes (reported by MS-Word used for editing this document). Therefore, each image takes about 16 KB with a representation that measures intensity of each small square in image as a pixel with an 8-bit integer, sets this measurement as a coordinate of a vector in a 150,000-dimensional vector space. The coordinates are organized the same way that a 500x300 matrix is represented by a column vector, namely, starting from entry (1,1) as the first coordinate, proceeding row by row to add the rest of coordinates, ending with matrix entry (500,300) as its 150,000-th coordinate. Clearly, such a representation is quite expensive, and intuitively

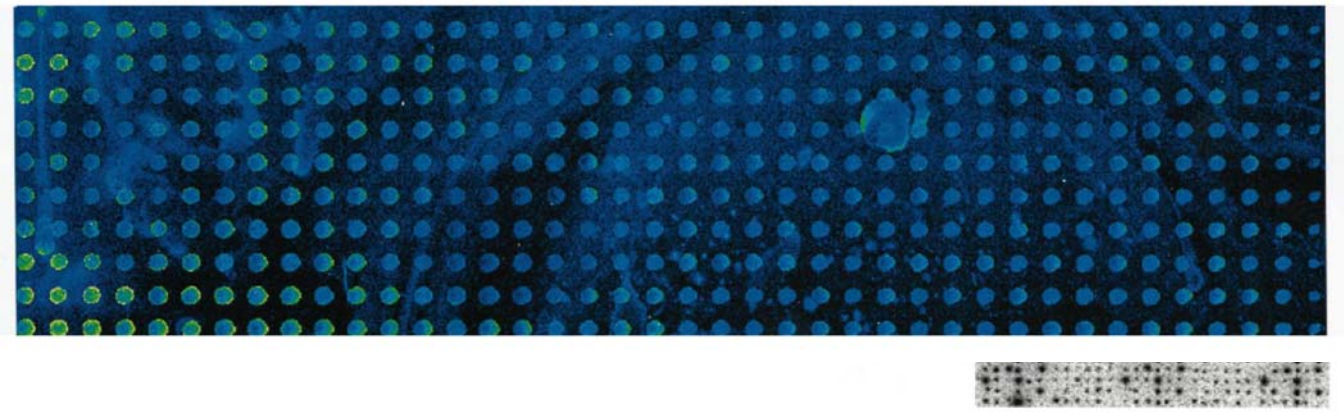
redundant. The image compression technology tries to provide alternative more efficient representations to reduce the storage requirement.

It is useful to remark that image compression, and in general, data compression could apply to a custom-collection of data as reference, say a carefully selected set of sample data **S** from the original data set. Once a good measure of compression is achieved for **S** , then one "generalizes" the application of the algorithm to the entire data set **D**. The quality of such a compression depends largely on the selected sample set **S** in terms of how representative it is, and the nature of data. For instance, one can achieve a far better compression quality when the images are from bacteria (as in page 1) rather then images of complex city scenes, people, or actions. We shall see that uniformity in shapes of image samples will have a great effect on economy of representation. The PCA is one method to achieve image compression using a sufficiently representative sample of a large data set. Its quality greatly improves as the individual images share higher and higher degrees of shape-similarity, or statistical correlation.

Example 1-Supplement. We provide another example from image processing, where the natural intuition that we have about faces and natural visual scenes is absent. This example illustrates the value of abstract/formal approaches to image processing, even when our visual perception provides part of the intuition. In analysis of microarray data, one must overcome a number of subtle data analysis challenges, from acquisition of raw signals to the final steps of statistical inference. A key step is to "preprocess" florescence intensity images from microarray slides for pattern recognition. The term "preprocessing" is subject to different interpretation under different circumstances, meaning remaoval of artifacts, noise, or otherwise image components and characteristics that hides essential features, or contaminates the data. Here, one important step of preprocessing is to decide how to assign a numerical value to intensity at each "dot" (site of molecular-level gene expression or other desirable activity to be elicited). Dots are patches of color recorded by the instruments, and they have quite variable looks and shapes, as in the images below.







In this example, we can use PCA to find an objective method for evaluation of intensity, and assigning numerical values to the dots in a microarray slide. This will give a more scientifically accurate representation of the desired features in each data element. While image compression is not necessarily the objective in this example, one will naturally achieve a remarkable compression level without loosing information, provided the PCA step is performed correctly adapted to the data set.

Metric Structures On Data Spaces

Consider once more the data set **D** as a subset of points in the space V of all possible such data types. Once the measurements/observations reported in **D** are given in terms of real or complex numbers, the space V will be given the algebraic structure of a vector space over the corresponding field of scalars. Given two data points **X** and **Y**, how should we measure their distance so that it reflects their similarities or differences? This question should not be addressed at only two samples of data, since the same rule must apply to every point in the set **D**. Thus, we might begin with a statistical measure of similarity, namely, correlation. The first statistical invariant of **D**, namely *mean(***D***)* = $\mathbf{X}_{\mathbf{mean}}$ is algebraic, and we do not need any metric structure to define it. The second invariant is the *variance* of **D**, *variance(***D***)* = σ , and we use the common measure of distance to define it. In terms of coordinates of **X** = $(x_1, x_2, \ldots, x_n)$ and **Y** = $(y_1, y_2, \ldots, y_n)$, $\sigma = \sum_{j=1}^{j=m} \| x_j - y_j \|^2$. It follows that $< X, Y >= \sum_{j=1}^{j=m} x_j y_j$ is the unique formula for the inner product on V whose associated distance function

(metric) is given by the formula for the variance $dist(\mathbf{X}, \mathbf{Y}) = \sum_{j=1}^{j=m} \| x_j - y_j \|^2$. The relationship between statistical invariants and geometric structure on V is firmly established by this simple observation.

When the data set **D** consists of arrays of complex numbers, the vector space containing the data is a complex vector space, so as to accommodate full advantage of using complex scalars. The mean of **D** is the algebraic average, as in the case of real scalars. But the formula for the variance has the hidden features for a *Hermitian inner product*, as we see below:

$\sigma = \frac{1}{m} \sum_{j=1}^{j=m} \| x_j - y_j \|^2 = \frac{1}{m} \sum_{j=1}^{j=m} (x_j - y_j)\overline{(x_j - y_j)}$, so that the associated inner product is given by the formula: $< X, Y >= \sum_{j=1}^{j=m} x_j \overline{y_j}$. Therefore, the inner product <…,…> is complex linear in the first factor and complex anti-linear (or conjugate-linear) in the second factor.

Data compression generally refers to the body of methods for representation of data **D** in the most economical way while keeping a certain level of fidelity upon retrieval of the transformed data T(**D**). A practical method to computationally model most data representation problems, including data compression, is to represent first both **D** and T(**D**) as subsets of the *same* vector space V that has also a metric as part of its structure, e.g. an n-dimensional Euclidean space with its standard formula for Euclidean metric given by the distance along the straight lines between pairs of points. The coordinates of a point **P** in V are called its *features*. If the data is given as *arrays of complex numbers*, then we select V to be a complex vector space together with the *Hermitian inner product*. The usual popular method to define PCA by maximizing the variance has a generalization to the complex data case, and we shall discuss it below. As we shall see later, the naive definition of PCA from *singular value decomposition* (SVD) of matrix theory is unsuitable for most naturally posed nonlinear generalizations of PCA. Thus, we should reformulate the problem in terms of concepts that bypass such difficulties and generalize with ease to the cases above. The key to such a new view point is to regard data sets as geometric objects, and derive their statistical invariants in terms of the underlying continuous, or complex analytic or complex algebraic geometric structures. To fit this intuition into a statistical context, we think of a geometric structure $\mathbf{\Gamma}$ that approximates a data set D in V as the geometric parameterization of D, and that geometric invariants of $\mathbf{\Gamma}$ are best approximations of statistical invariants of D.

One way to formulate the data compression problem is to introduce a parameter ε that, roughly speaking, plays the role of error tolerance in the model that has used the transformed version T(D) instead of

using the original data D. Call a data point (that is, a *typical member of* D) and the result of its transformation, by analogy **P** and T(**P**) (hopefully also a *the typical member* of T(**D**).) When the distance in V between **P** and T(**P**) falls within ε, then we have fulfilled the error-tolerance requirement of the problem for the typical data elements. A useful way to interpret the notion of a *"typical data point"* is via average statistics with respect to all data points. The simplest such statistical interpretation is to use the average or mean of distances between **P** and T(**P**). This average is called *the reconstruction error* for the transformation T applied to the data set D. *The objective of any data representation/transformation is to minimize the reconstruction error by a judicious choice of T.* Once an error tolerance ε is given, then any T whose reconstruction error is less than ε would do the job. However, data representation problems may not have a well-defined error tolerance given *a priori*. Rather, the problem usually gives an indication of only a *"worst scenario error"*, leaving the selection of the *"best possible error"* to the modeler. In such cases, we must actually solve a more serious optimization problem: *Find the transformation T such that the reconstruction error is the minimum allowable subject to the requirements of the representation* (recall the band-width restriction, etc in the above-mentioned example from multi-media.) A solution to this type of optimization problem is usually controlled by the source and the nature of data. In perceptual systems, the external stimulus impresses upon the system in such a way that both the scale of the stimuli and the resolution of the system intertwine and give rise to a multi-scale representation of the stimuli, eventually taking the final form via higher-level cognitive processes. Thus, multi-scale analysis of data could very well fall into such applications, and we face the question of formulating a *multiple-scale, multi-resolution generalization of PCA* and its closely related concept of ICA (Independent Component Analysis). Here we wish to remark that doing PCA in a hierarchy of scales *is not the same* as the multi-scale PCA intended here. Mainly, there are computational subtleties that reflect the result of interaction between scales, so that in our definition of multi-scale analysis, there could be couplings between the sets of equations governing different scales. To accommodate the couplings, one must introduce geometric structures that integrate the structures at different scales naturally, and lend themselves to the definition of optimization problems in systems with coupled constraints. We shall illustrate below a general geometric construction to achieve multi-scale analysis.

What are the general options for the transformation T when the data set D is known (i.e. D is already given in terms of the feature coordinates of its points in the n-dimensional vector space V)? To begin with, we make the following simple but very useful assumption that is easily achieved by translating the data set to the point *mean(***D***)* = $\mathbf{X}_{\mathbf{mean}}$ that is computed as the mean of D in the original coordinate system..

Assumption. We arrange for the coordinate system in V to have the mean of the data set D as its origin. From now on, we can and will assume that all data sets that we study have mean equal to zero.

This basic simplification allows us to add a useful and statistically meaningful restriction on the family of transformations T, namely, that T(**0**) = **0**; or that *T preserves the mean of the data.*[1] This criterion seriously limits the geometric properties of mappings, from the admissible sub-collection in basic rigid motions, for example, excluding translations. The simplest transformations that satisfy T(**0**) = **0** are the linear transformations. Here, such a T is given by a linear transformation of V to itself, and in its given coordinate system, T is represented by an n×n matrix! The next simplest family is not so easy to define. Depending on the application and the nature of data, one could define the set of piecewise linear mappings (call the family PLM) as suitable family to be used in optimization, or, the set of algebraic transformations given by systems of polynomial equations (call the family SAE for *systems of algebraic equations*) if the combinatorial and discrete features are the dominant factors. The two collections PLM and SAE exhibit vastly different properties when one wishes to draw functions from them in approximation tasks. PLM fall essentially into the domain of differential topology and geometry, while SAE is best dealt with in terms of algebraic geometry.

Abstractly, we have to compute the reconstruction error for every choice of T, say f(T), using the data D and its transform T(D), and find the T that gives the minimum of the function f(T). Let us examine the simpler case of linear transformations. In terms of formulas, let $\left(x_{ij}\right)$ be the matrix representing T, and let the average error be defined below, in which the summation reflects the average over all distances of elements $p \in D$ to $T\left((p_j)\right) = \sum_j x_{ij} p_j$ as in:

$$f(T) = \frac{1}{\#(D)} \sum_p \sum_i \left(\sum_j x_{ij} p_j - p_i \right)^2 .$$

In principle, the problem of finding the coefficients $\left(x_{ij}\right)$ to minimize f could be handled using calculus. In particular, the space of n-by-n matrices where $\left(x_{ij}\right)$ live is the same as the Euclidean space of dimension n^2. In practice, the function f has *far too many parameters* and its minimum is not easy to find by calculus alone! A compromise is to limit the choices for the function T by requiring useful geometric restrictions that help simplify the calculation, as well as lend itself to effective algorithmic computation. As we shall see later, there are several powerful methods to approximate the minima of such functions with many variables, among them, neural networks, evolutionary and genetic computation are well studied and often provide outstanding results.

[1] The geometric interpretation of this property is that the mappings T preserve a preferred base-point, here, the origin.

Among all possible choices of n-by-n matrices, we are often interested in those transformations that decrease the size of the data set by reducing the dimensionality of the image T(D). The intuition for this point is that the given data set in V uses all coordinates of V as its defining parameters. If the image T(D) lies in a subspace W of lower dimension, then the coordinate system for W could be used as defining parameters for T(D), hence much more economically representing D as well. Thus, we are led to search for the best T whose rank is k, where k is a candidate for the dimension of the image T(D), typically, much less than dimension of D. Often, D itself lies in a hidden subspace of dimension usually less than N in almost all practical circumstances.

At this point, we have an abstract formulation[2] for the problem of data compression through reduction of dimensionality by linear transformations! Let us consider the simplest possible case: when k=1, and we wish to find T such that the image of T is a line and the reconstruction error is minimum. In statistics, this is related to the problem of finding *a linear regression* for the data D.[3] It turns out that this problem has an elegant and simple solution, called the Principal Component Analysis (PCA). In our multi-media example, one uses the PCA in order to obtain a compressed representation T(D) from D, which can be stored or transmitted in real-time. Principal Component Analysis (PCA) is closely related to Karhunen-Loeve transform and the Hotelling transform that are standard techniques in feature extraction and data compression. See, (Oja, 1983) and (Devijver and Kittler, 1982) for more details and historical remarks. As an example, take a sequence of 8 x 8 pixel windows from a digital image. They are first scanned into vectors **p** whose elements are the gray levels of the 64 pixels in the window. In general terms, the input vectors are random vectors **p** with N elements. No assumptions about the probability density of the vectors are needed, as long as their first- and second-order statistics are known or can be estimated from a sample. Typically the elements of **p** are measurements like pixel gray levels or values of a signal at different time instants. They are mutually correlated, therefore, represented with a large degree of statistical redundancy.

In the PCA transform, vector **p** is linearly transformed to another vector **q** with K elements, $K < N$, so that the redundancy induced by the correlations is removed. This transformation is achieved by finding a rotated coordinate system such that the elements of **p** in the new coordinates become uncorrelated.

---

[2] The more abstract formulation should not restrict the choice of T to linear transformations of V. Rather, all mappings of V that are measurable are theoretically acceptable. Nonetheless, assuming T to be continuous or differentiable, or even definable by a system of algebraic equations (i.e. an algebraic morphism in the sense of algebraic geometry) has a better chance of success, as general measurable transformations of V form a very large and complicated space, making the optimization problem impractical, if not impossible.

3 One must note that sometimes linear regression is formulated by minimizing another error function that is simpler than what we have considered as the reconstruction error.)

Example. Let D be a data set whose pdf could be estimated by a Gaussian, say **p**. Suppose that **p** has a Gaussian density that is constant over ellipsoidal surfaces. Then, the center of the ellipsoid is the mean for D, and the principal axes of the ellipsoid provide the directions for projection of data, providing maximal variance. Equivalently, to provide the best estimate for data in terms of minimum reconstruction error, one must choose the new rotated coordinate system that coincides with the principal axes of the ellipsoid. In addition to achieving uncorrelated components, the variances of the elements of **q** will be decreasing in most applications, with a considerable number of the elements so small that they can be discarded altogether. The elements that remain constitute the vector **q**.

computation of principal components

Suppose we have 20 column vectors (e.g. faces) that have dimension 10,000 (for example from a 100x100 pixels image in gray-scale.) Call these F1, F2, …F20. Now, we make A=[F1, F2, …F20], size(A')=20x10,000. Calculate C=A'*A (and theoretically consider) M=A*A'. Size(C)= 20x20 and size(M)=10,000x10,000. It is a bad idea to work with M, since it requires a huge amount of memory. Fortunately, eig(C) gives the first 20 largest eig(M), so no need to use M for that. Let us assume that all eigenvalues are non-zero, or equivalently, rank(A)=20, or equivalently, the 20 vectors for these faces are linearly independent, Even this may not be true, I wish to comment that if you select randomly 20 100x100 faces and make them into vectors, they are almost always linearly independent! You must truly look hard to find a face that is a linear combination of several randomly selected faces, unless you cheat and you set it as such a linear combination!

It is good to look at M for theoretical computation, since M transforms each data vector Fj to another face vector of the same size. But, C does not act on Fj, since the sizes don't match.

So, where does C act? Consider the column space of A and let W = span{F1,…Fj}. Dim(W) is at most 20, and rank of A is dim(W). Note that vectors in W are 10,000-dimensional, but there are very few of them. So, they occupy a very small portion of the space R^10,000. Consider V = rowspace(A). The dim(V)=20=rank(A) also, since a theorem of linear algebra says rank(A)=dim(W)=dim(V), either way. So, dimV=dimW,
That says these two vector spaces are isomorphic. That is, they both look like the same Euclidean vector space. The difference is that vectors in V are 20-dimensional.

When we do PCA, we find 20 eigenvalues and correspondingly 20 eigenvectors. These eigenvectors naturally live in the space on which C operates, namely, V. And of course, we see that the eigenvectors are also given by 20 dimensional vectors. Call the eigenvectors {v1, v2, …, v20} (living in V). Cal the theoretical eigenvectors of M, {w1, w2, ..w20} (living in W). Either one of these must have unit length, by our choice. Each set,

therefore, is an orthonormal basis for the vector space they live in. Both sets could be used for approximation of our data. The trouble is: we cannot easily plot the image of the 20-dimensional vectors as faces, and, from there, go on with the application of face recognition etc. But each one of the 10,000 eigenvectors can be easily made into a face!

Just use the command reshape (m1, 100, 100) and you have an array of the right size, that can be then represented as an image. Since the length of m1 is 1, this image may be too dark or too light, depending how close the average length of each F1,… is to unit. Anyhow, if you type in G1=(sum(F1.*m1))*m1, you get a 10,000 dimensional vector that is the component of the face F in the direction of the first principal component. If you reshape that one and plot it, you get a great face image! After these preliminary discussion, let us see how we can transform the 20-dimensional eigenvectors that we have calculated, to the 10,000 dimensional eigenvectors of M that we have not calculated because of the size of the matrix M.

Note, size(v1)=20x1, has 20 rows, since it is a column vector. Try u1=A*(v1) , …. u20=A*(v20). size(u1)=size((10,000x20)*(20x1))=10,000x1 , so we are on the right track and have a vector in the right vector space. Also, note that A*(v1) is simply a linear combination of columns of A using the entries of v1 as the coefficients:

If $v1 = (\beta_1 , \beta_2, \ldots, \beta_{20})$, then, $A*(v1) = \beta_1*F1 + \beta_2*F2 + \ldots + \beta_{20}*F20$. The conclusion is that u1 lies in the subspace W spanned by the data vectors F1,…F20. Similarly for the other u2,.., u20. The only thing that remains is to check that u1,.. are eigenvectors of M. For now, suppose C*(v1) = c*(v1), where c is an eigenvalue, and let's do the matrix multiplications, remembering the rules for the transpose of matrix products:

C*(v1) = c*(v1) , so A'*A*(v1) = c*(v1) ; multiply both sides of the last equation by A to obtain: A*A'*A(v1) = c*(A*(v1)) . Rearrange the parentheses and let M = A*A' and A*(v1) = u1 to get:

M*(u1) = c*(u1), meaning that u1 is the eigenvector of unit length corresponding to the eigenvalue c, just as v1 is the unit eigenvector of C for the same eigenvalue c. This is true for all u1, u2, … and all corresponding eigenvalues c1, c2, … We conclude that {u1, u2, …u20} is an orthonormal basis for the subspace W spanned by the data vectors, and they are eigenvectors for the correlation matrix M = A'*A. With respect to this basis, we can rewrite the original data:

Gj = <Fj , u1>*u1 is the first component of the data Fj , and it has the same dimension as Fj , so we can use reshape(…) command and make it into a 100x100 matrix, that is, the pixel values of the appropriate image. The reconstruction error for this estimate is:

Ej = Fj – Gj = Fj - <Fj , u1>*u1 which is also essentially a 100x100 matrix of pixel values of a faint image, as you could check for yourselves.

Probability Theoretic Formulation

In the context of probability/statistics, assume that the elements of the data set D is represented by column vectors X in an m-dimensional real vector space with zero mean. Let the desired transformation T be represented in the standard coordinates by a matrix written in block-form $W = [W^1, W^2, ..., W^m]$, where each $W^j = [w_1^j, ..., w_k^j, ..., w_m^j]^T$ is a column of the matrix W. Consider the formula for T provided by the linear combination $Y^1 = \sum_{k=1}^{k=m} w_k^1 X_k = (W^1)^T X$, and subject to the condition of keeping the length of W^1 constant (say unit length for simplicity), while demand that the variance of the random variable Y^1 be maximal among all possible choices of such vectors W^1. Geometrically, keeping the length of W^1 constant is the same as searching among the vectors forming the surface of the sphere of radius one for the desired coefficient vector, and trying to maximize the expected value

$$E\{\langle Y^1, Y^1 \rangle\} = E\{\langle (W^1)^T X, W^1 X \rangle)^2\} = (W^1)^T C W^1$$

of the random variable Y^1 given by the formula above. Recall that $C = E\{\langle X, X \rangle\}$ is the correlation matrix, and $E\{...\}$ is the expectation operator with respect to the density of the input vectors from D, that is, the weighted average that could be (naively) approximated using histograms of projection of data on each standard axis. The vector Y^1 yields simply the coordinate of the data point X along the first principal direction. To find other principal directions, we proceed inductively: In inductive step j+1, we require the variance of Y^{j+1} to be maximized, the length of the vector W be constant (say unit length), and that W^{j+1} be orthogonal to the previously calculated weight vectors $W^1, ..., W^j$. To use the closed-form solutions given here, the eigenvectors of the covariance matrix C must be known. These are rarely known in practice. In an on-line data-compression application like image or speech coding, it is usually not possible to solve the eigenvector-eigenvalue problem for computational reasons. The PCA solution is then replaced by sub-optimal standard transformations. Another alternative is to derive gradient ascent algorithms that converge to the solutions of the problems, i.e., to the eigenvectors. This approach is the basis of the neural network learning rules., to be discussed later.