Chapter 10

The Principle Components of an Image

Eigenvectors, eigenvalues, svd, introduction to principle component analysis, dimensionality reduction, image compression, image matching

10.1 Introduction

We ended the last lecture (on sampling) with the amazing observation that a continuous function (which, because it "lives" on the continuum is infinite-dimensional) could be represented by a finite set of number (the Fourier coefficients). This is an example of DIMENSIONALITY REDUCTION, or the idea that the basic structure of a set of data can be captured in fewer dimensions than the data itself. It works for sampling because the "information" is really concentrated within the Nyquist band – not all frequencies.

We considered different frameworks for matching images in the previous two lectures, and we now build on this to get more of a MACHINE LEARNING perspective. In Lecture **??** we introduced the idea of a space of images (see Figs. **??**; within this space there were different ways of "projecting" one image onto another via the correlation operation. (Remember: different metrics). In the next lecture 12 we put a probability stucture on images and introduce Bayes theorem as a mechanism to guide the "best" classification decision. In this lecture we start the process.

Here is the basic issue that guides this lecture: image correlation is expensive, and many of the pixels are highly correlated with one another. Is there a way to reduce the dimensionality of 'pixel space?' The answer, it turns out, is a beautiful one and involves a central result from linear algebra. We'll try to motivate it but suggest that, eventually, you take a full course in linear algebra to fully understand it.

The second question is whether the projection idea captures all that there is to matching. Clearly this is not the case.

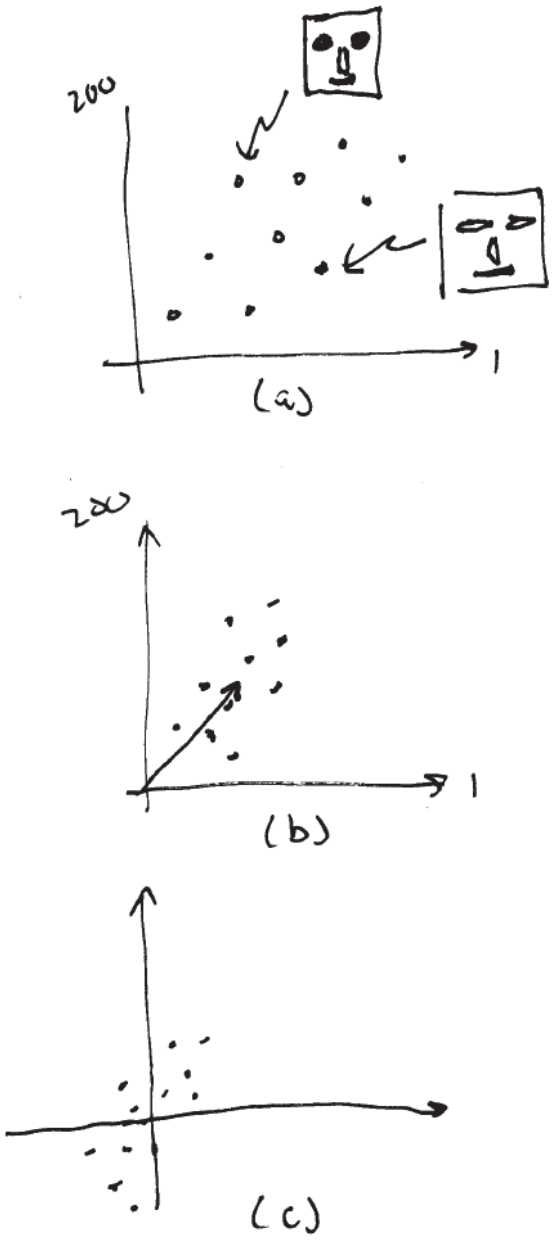

Figure 10.1: A group of images plotted in pixel-space. (a) Two example images are shown, but there is a slightly different image for every point. (b) The mean vector points to the average of the set of images. (c) Subtracting the mean from each image *centers* the data around the origin. Note values now average to 0.

Figure 10.2: (top) Dataset of faces (from Drexel example). (bottom) Mean of the dataset obtained by averaging the pixel brightness for each pixel separately.

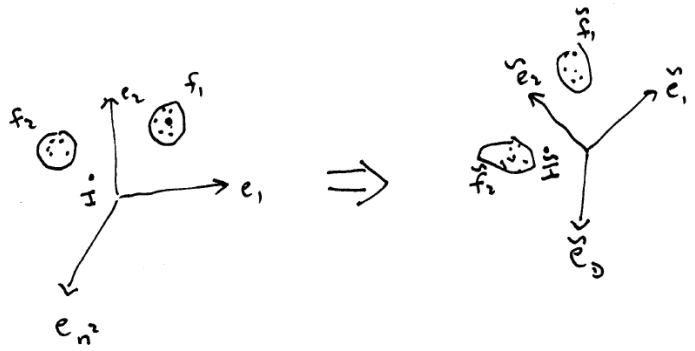

Figure 10.3: Dimensionality reduction for spaces of images. An $(n \times n)$ image naturally lives in an n^2-dimensional space, which can be quite large. The idea of dimensionality reduction is to find another (hopefully) lower-dimensional space of the same type, i.e. one in which the new dimensions are functions of the original dimensions. Here we illustrate the idea. An image of an unknown face, I, needs to be classified as either face-1 (F_1) or face-2 (F_2). If this could be done in a lower-dimensional space, e.g. by the projection (e.g. correlation) scheme for classification developed in the last lecture, the process would be much more efficient.

10.1.1 Is dimensionality reduction even feasible?

How should we think about dimensionality reduction? To start, we have a cloud of points in a high-dimensional space (for a (100 x 100) image this would be 10,000 dimensions); are they all equally informative? Probably not. Recalling the templates, pixels in the background are less interesting than those on the face; or those that might separate an "H" from a "Y." Here's one idea: let's just choose, say, 50% of the pixels at random – will this work? Amazingly, the answer is Yes in the following sense: for a collection of images in this (inter-image) distances about the same (in proportion). This is the celebrated Johnson-Lindenstrauss theorem (about which more in the advanced course).

10.1.2 Building Intuition about 'Principle'

Our goal here is to introduce the concept of Principle Components Analysis (or PCA), and we start with a physical observation. (This also motivates where the word 'principle' comes from.)

Some objects are easier to spin than others. Just pick up a sphere, for example, and it will spin fine. Now, pick up a top: it will spin if you hold the "handle" and put the point on the table, but unlike the sphere, this axis is special – a *principle axis* of inertia. Now, pick up a potato and spin it ... Irregular objects will mostly wobble (their motion is a complicated 'sum' of the motions of each 'molecule' of

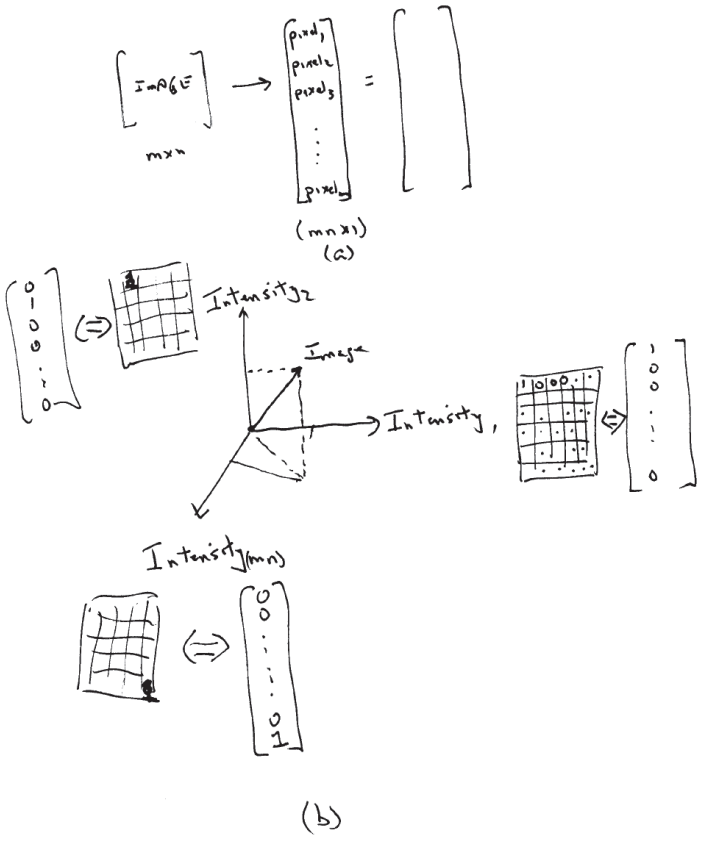

Figure 10.4: The standard basis in the space of images. (a) Think of an image as an $m \times n$ matrix of intensities. Stacking each row on top of the next yields an $mn \times 1$ vector. (b) Plot these vectors as points in an mn-dimensional space, where each axis indicates an individual pixel. Shown are the first, second, and mn^{th} directions.

potato around the axis). But a proper rotation of the potato may work. This is the idea here: we will need to "rotate" the axes in our spaces of images to find those that are principle.

10.1.3 Building Intuition about Image (Vector) Spaces

Now, let's review the model of an image as a point in a space of images (recall Figs. 7.10 and 7.11). If we had an image and "vectorized" it, this simply transforms a matrix of size $(m \times n)$ into a vector of size mn by rotating each row of the image into a column vector and then placing these columns on top of each other; see Fig. 10.4a.

This mn-dimensional vector can then be plotted in an mn-dimensional vector space. A (real) VECTOR SPACE is a set of vectors on which two operations are defined: vector addition and multiplication by a (real) number. These operations imply that

if we add two vectors in the space we obtain another vector that is also in the space. Or if we multiply a vector by a number we get another vector (with different length) that is also in the space. (If we think of 0 as a number, what does this imply? The 0-vector is in the space.)

Because of the equivalence that we set up between images (matrices) and vectors, it follows that we could also have thought of the vector space of $(m \times n)$ matrices. (This helps a little in thinking about the role of abstraction in mathematics.)

Now, let's be a little more careful about thinking of the vector space in which we're representing images. To do calculations in this space, we have to determine in which directions the coordinates are pointing. Here we set it up so that one could think of each axis as a unit vector "indicating" a single pixel. There are just enough of them to represent each of the pixels. Each pixel's intensity is then the projection of the image onto this. Since the pixels are in different positions these vectors are "independent." It's natural to think of these as column vectors with a 1 indicating the position of the pixel i and a 0 elsewhere. Symbolically,

$$\vec{e_i} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 1 \\ \vdots \\ 0 \\ 0 \end{bmatrix} \tag{10.1}$$

A BASIS is a set of vectors that are INDEPENDENT and that SPAN the space. Independence means that there is no linear relationship among them and span means that any vector in the space can be expressed in terms of them. In a 3-dimensional vector space, for example, three basis vectors are required and they cannot lie in the same plane. Any other vector is a linear combination of these three basis vectors.

Our image (Fig. 10.4b) can thus be expressed:

$$I = \sum_{=1}^{mn} p_i \vec{e_i}. \tag{10.2}$$

Is this the only basis for this space? Is this the "best" basis? We'll be working in this lecture to find another one and a suitable notion of "best."

We now switch from an individual image to a collection of images on our way to asking: which is the best basis to describe the collection of images? Three main considerations will arise: (i) how are these distributed and (ii) what will we be doing with them? The third consideration (iii) reducing the number of computations, we'll deal with later in the lecture.

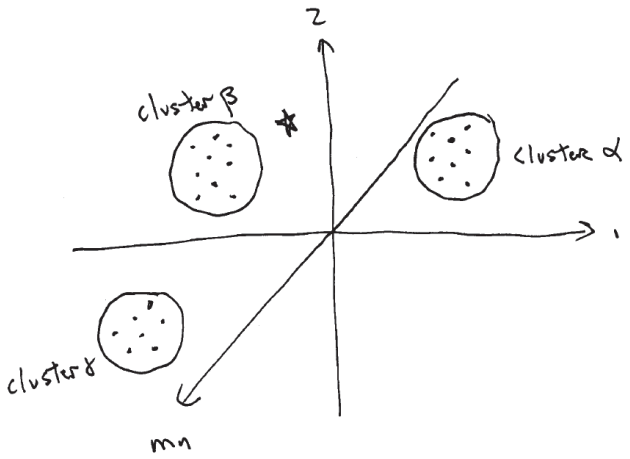

Figure 10.5: The geometry of classification in image space. Suppose we have been given a set of images taken from different individuals (call them α, β, γ). In the ideal, well-structured case, the subset of images from each individual forms a distinct cluster, separate from the others. (Geometrically, this means that the 'spread' within each cluster is 'small' with regard to the 'distance' between the clusters.) Now, given a previously unseen image ($\star$), it is natural to classify it as belonging to the closest cluster.

10.1.4 Classification in Image Space

One task with images that we've looked at for the past three lectures is classification. Suppose we were given a set of images for each class of objects of interest. (We've been thinking about faces.[1] Now, given an image of an unknown person (or category of objects), find that group of known images to which it "most likely" belongs. First we did this with correlation and then with Bayes theorem. Here wer plot it in image space (Fig. 10.5. The idea is to first build a set of clusters in image space, one for each "individual." Hopefully these clusters are smaller in diameter than the distance between them. Now, when a previously unseen image $\star$ of one of these individuals comes along it can be associated with the "nearest" cluster. (Since the unseen image captures directly the APPEARANCE of the individual, it will vary between photographs; thus one wouldn't expect it to overlap exactly with a previously acquired image.)

In this lecture we will not tackle the full recognition problem, which has many other aspects. For example, there are computational advantages to tesselating space

[1]Face recognition by computer is an enormous field; these introductory remarks barely set the stage and certainly do not give a sense of the commercial possibilities. See e.g `www.face-rec.org/` for some entry points into this literature.)

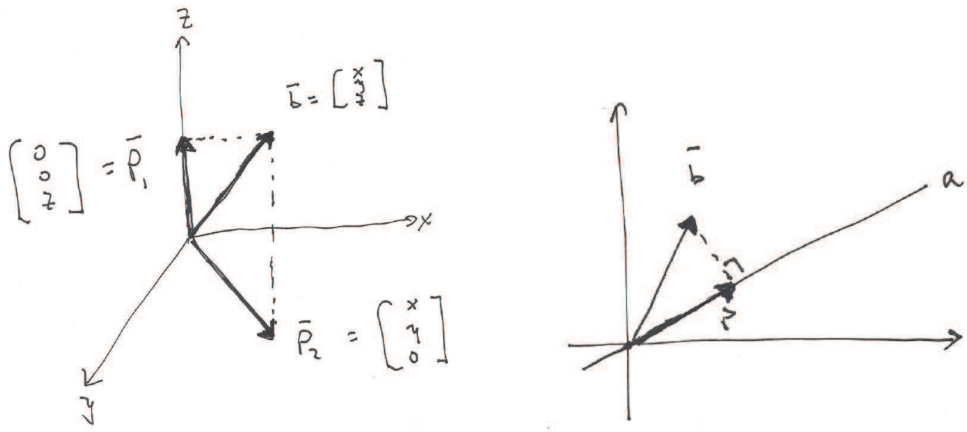

Figure 10.6: The geometry of projection in vector spaces. (left) Projection of the vector $\vec{b} = [x\ y\ z]^T$ onto the z-axis or onto the (x, y)-plane. (right) Projecting the vector b onto the line a. Figs from Strang, p206

rather than searching over distances. That is, we could draw boundaries between the clusters, and then classification amounts to determining on which side of the boundary a point fall. Design of classifiers will be treated in more detail next term. For now we focus on description of a cluster. (Yes, don't worry, we'll return to classification issues at the end.)

10.1.5 Warm-up: The Projection Operation

In this section we review some of the background on projections and sub-spaces; see Fig. 10.6. It's a sort of warm-up for what's to come, because several important structures emerge directly.

The basic idea is to start thinking about the relationship between vectors and projections of those vectors onto either other vectors (a 1-D subspace) or planes (a 2-D subspace), etc. This will be a form of decomposition. After we discuss it, we'll talk about putting the pieces back together again.

We begin, as alway, by thinking of about the form:

$$\textbf{[vector]} = \textbf{[matrix]}\ \textbf{[vector]}.$$

Now, we first ask what this matrix is for the projection of a vector onto one of the axes (Fig. 10.6 left). Clearly, if we start with $\vec{b} = [x\ y\ z]^T$ and project it onto the z-axis, we readily compute (following Strang, p 206):

$$\vec{p_1} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ z \end{bmatrix}. \tag{10.3}$$

Now, to project onto the (x, y)-plane, we similiarly get:

$$\vec{p_2} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} x \\ y \\ 0 \end{bmatrix}. \tag{10.4}$$

How might these projections be found? (See Fig. 10.6(right). Suppose we wish to project a vector $\vec{b}$ onto a line a; we need to specify what the closest point $\vec{p}$ is on a. Let's write this as $\vec{p} = \alpha\vec{a}$ where α is a scalar. This is what we have to solve for.

Now, looking at the figure, we immediately notice that the error (the dotted line) $\vec{e} = \vec{b} - \vec{p}$ has to be perpendicular to $\vec{a}$ (just rotate the dotted line around a point of attachment ($\vec{b}$) to see this.) So,

$$\vec{e} \cdot \vec{a} = 0 \Rightarrow \vec{a} \cdot (\vec{b} - \alpha\vec{a}) = 0 \Rightarrow \vec{a} \cdot \vec{b} - \alpha\vec{a} \cdot \vec{a} = 0$$

or

$$\alpha = \frac{\vec{a} \cdot \vec{b}}{\vec{a} \cdot \vec{a}} = \frac{\vec{a}^T\vec{b}}{\vec{a}^T\vec{a}}$$

A few comments about these operations on vectors that will prefigure the concept we're working toward (eigenvectors and eigenvalues). First, note that if we apply the first matrix again, we get the same thing:

$$\vec{p_1} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \vec{p_1} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \vec{p_1} \tag{10.5}$$

That is, the composition of a projection with itself is just the projection. (This is the IDEMPOTENT property.) Second, we can put certain scalars into these equations:

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \vec{p_1} = 1\ \vec{p_1} \tag{10.6}$$

and

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \vec{p_2} = 0\ \vec{p_2} \tag{10.7}$$

That is, certain vectors, when acted upon by a matrix, return that vector scaled in length. (This can mean shrink to 0 or leave the vector completely unchanged.) We have seen this property before (think sinusoids and linear systems.)

Finally, we observe the recombination:

$$\vec{p_1} + \vec{p_2} = \vec{b}. \tag{10.8}$$

In other words, we can view the projections from two 'perspectives.' From the 'vector' perspective, it is built from components. From the 'vector space' perspective, these parts 'live' in different subspaces of the full space. Combining the subspaces (in the example above, the 1-D and the 2-D spaces considered) yields the total 3-D space.

10.2 PCA

We now start the central topic of this lecture. Our goal is to represent the data not necessarily in standard coordinates but in ones "driven by" the data. Let's start by finding the first coordinate (this repeats a little material from two lectures ago, but it's nice to be reminded.

10.2.1 Best approximating line

First, let's move the data around the origin (subtracting the mean); then we can ask *which is the best line passing through it*; see Fig. 10.7. This requires a definition of best.

Let's conceptualize the best line as w, with a unit vector $\vec{w}$ representing it. Now, think of the data points (images) also as vectors. The projection of a typical data point x_i (the i-th image) onto this line is a vector

$$\underbrace{(\vec{x_i} \cdot \vec{w})}_{\text{magnitude}} \underbrace{\vec{w}}_{\text{direction}}$$

so the difference, or residual, is $||\vec{x_i} - (\vec{x_i} \cdot \vec{w})\vec{w}||^2$. So, we want to find the "best" vector $\vec{w}$ that is the:

$$\arg\min_{\vec{w}, \|\vec{w}\|=1} \sum_{i=1}^{n} \|\vec{x}_i - (\vec{x}_i \cdot \vec{w})\vec{w}\|^2,$$

where we represent our line as a unit length direction vector $\vec{w}$. We can expand the summand:

$$\begin{aligned} \|\vec{x}_i - (\vec{x}_i \cdot \vec{w})\vec{w}\|^2 &= (\vec{x}_i - (\vec{x}_i \cdot \vec{w})\vec{w}) \cdot (\vec{x}_i - (\vec{x}_i \cdot \vec{w})\vec{w}) \\ &= \|\vec{x}_i\|^2 - 2(\vec{x}_i \cdot \vec{w})^2 + (\vec{x}_i \cdot \vec{w})^2 \\ &= \|\vec{x}_i\|^2 - (\vec{x}_i \cdot \vec{w})^2. \end{aligned}$$

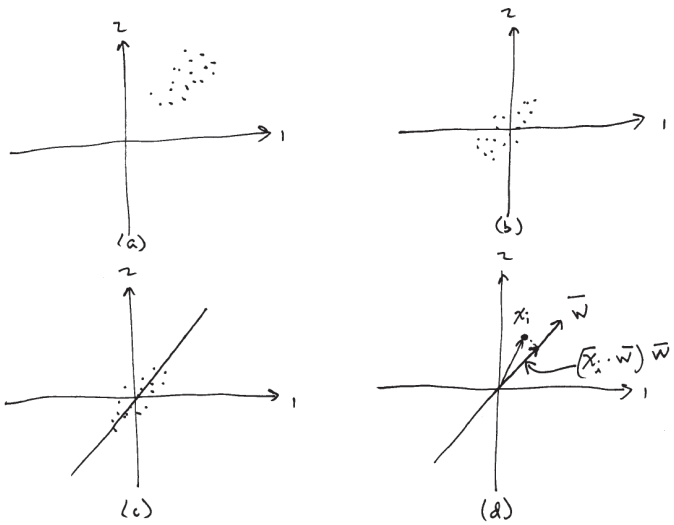

Figure 10.7: Defining a *best* line passing through a collection of data points. (a) The data points in mn-dimensional space (for simplicity, only 2 dimensions are shown). (b) Subtracting the mean moves the cluster of data points to the origin. (c) A natural best line through the data points. (d) The sense in which the lines is best. Think of each datapoint as a vector and project it onto the line. We seek that line with the largest total projecton.

From this we can conclude that minimizing the squared distance of the observations from the line is equivalent to maximizing the squares of the projections of the observations onto $\vec{w}$.

This notion can be expressed concisely in matrix notation. First, let's build up a giant DATA MATRIX, X, in which the columns are the different data samples (remember, each of these is a vector, x_i:

$$X = \begin{bmatrix} | & | & & | \\ x_1 & x_2 & \cdots & x_n \\ | & | & & | \\ | & | & & | \end{bmatrix}$$

Thus our problem becomes: find $\vec{w}$ such that:

$$\arg\max_{\vec{w},\|\vec{w}\|=1} \left\| X^{\mathsf{T}}\vec{w} \right\|.$$

This seems really hard; in fact, it's not even clear how to think about this problem. Trying every line seems crazy. But clearly it has something to do with X – for different data one would expect different answers. The trick is to think of the matrix X as a transformation (an operator that takese vectors as input and gives a new vector). We'll develop this now, via an algebraic detour.

Caution: Note that X is an (dxn) matrix: each datum is a vector of length d (say, d pixels) with n columns (samples). In general, of course, $d \neq n$ so the matrix is not square. Keep this in mind until we get through the next section.

10.2.2 Eigenvectors and Eigenvalues

Earlier we examined the operation of applying a matrix to a vector: it rotates and rescales it (let's ignore translation, for now). (If the new vector is also going to be in this vector space are there other possibilities?) We have also encountered some special vectors that are only scaled by the operation of the matrix – the eigenvectors. (Note: these eigenvectors will, natually, depend on the matrix.) They enjoy the special property that, for a matrix A, the eigenvectors $\vec{x}$ and eigenvalues λ:

$$A\vec{x} = \lambda\vec{x} \tag{10.9}$$

A geometric view

To be concrete, consider at an example from Banchoff and Wermer (p 76): reflecting vectors around a line L passing through the origin Fig. 10.8. In general, defining this matrix might seem pretty complicated; in any case, it would be some (2 x 2 matrix. However the construction in the figure shows us that it's more natural to work in the $(L, L^{\perp})$ coordinates than in (x_1, x_2) coordinates. For vectors in direction L there's

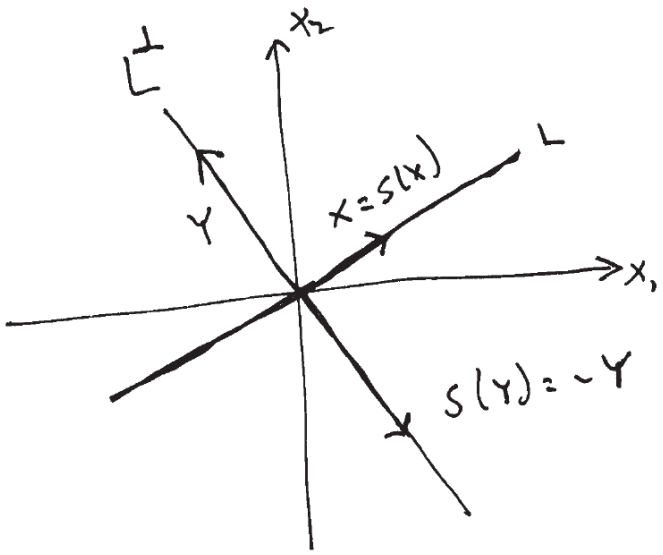

Figure 10.8: Certain coordinate frames are privileged. Example: Reflection of a vector around the line L by the operator S. If a vector X lies on the line, then $S(X) = X$. If a vector lies perpendicular to L, then $S(Y) = -Y$. This suggests a very natural coordinate frame, $(L, L^{\perp})$, a rotation of the original (x_1, x_2) coordinates, in which to work.

no change, while for vectors in direction $L^{\perp}$) we have to flip them. Any vector could be described in these coordinates. Specifically, since these vectors are unchanged except for the scalars, we observe: For eigenvectors in direction L we have $\lambda = 1$ and for eigenvectors in direction $L^{\perp}$) we have $\lambda = -1$. In effect to characterize this operation for any vector it was helpful to work in $(L, L^{\perp})$ coordinates rather than in (x_1, x_2) coordinates. So, we're thinking, to reflect a vector, first express it in these $(L, L^{\perp})$ coordinates, then scale it, then "re-express" it into the original coordinates (if needed).

Another point to emphasize: we have two bases for the example in Fig. 10.8: (x_1, x_2) and $(L, L^{\perp})$. Both consist of two orthonormal vectors and any vector can be expressed in each of them. The coordinates will be different – depending on the basis – but the vector will not. At the center is the observation: $(L, L^{\perp})$ is just a rotation of (x_1, x_2). Thus the above 're-expression' idea is: rotate - scale - unrotate. This sequence is fundamental.

Aside: the standard way to find eigenvalues/vectors

It's instructive to include a textbook example of how this material is normally introduced. (Most students in class will have seen this before, because it's introduced in many courses related to the nullspace and solving of systems of linear equations). This provides a nice starting point.

We begin by summarizing a Strang remark:

- *eigenvalues are about the invertability of a matrix;*
- *eigenvectors are about diagonalizability of a matrix.*

Suppose we seek the eigenvalues $\lambda_i, i = 1, 2..., n$, of a matrix A. Since it's eigenvectors $\vec{x_i}$ specify the NULLSPACE of $(A - \lambda I)$, where I is the identity matrix, we have

$$\text{If}(A - \lambda I)\vec{x} = 0$$

has a non-zero solution, then $(A - \lambda I)$ is not-invertible or $\det(A - \lambda I) = 0$. Thus we have the main result:

$$\lambda \text{ is an eigenvalue of } A \text{ if and only if } \det(A - \lambda I) = 0.$$

This suggests a two step recipe:

1. Find eigenvalues:
   (a) compute $\det(A - \lambda I) = 0$ and find roots of the resulting polynomial.
2. Find eigenvectors:
   (a) For each λ, solve $(A - \lambda I)\vec{x} = 0$.

Here's an example from Strang. Let

$$A = \begin{bmatrix} 1 & 2 \\ 2 & 4 \end{bmatrix}$$

then

$$A - \lambda I = \begin{bmatrix} 1 - \lambda & 2 \\ 2 & 4 - \lambda \end{bmatrix}$$

so we get the det() equation and set it to 0:

$$(1 - \lambda)(4 - \lambda) - (2)(2) = \lambda^2 - 5\lambda = \lambda(\lambda - 5) = 0$$

which has two roots, $\lambda = \{0, 5\}$. Following the recipe (denote $\vec{x} = [a\ b]^T$), we get for $\lambda_1 = 0$:

$$A - 0I)\vec{x} = \begin{bmatrix} 1 & 2 \\ 2 & 4 \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

which gives the eigenvector $[a\ b]^T = [2\ -1]^T$.

Now, for the other eigenvalue $\lambda_2 = 5$, we have

$$(A - 5I)\vec{x} = 0 \Rightarrow \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \end{bmatrix}.$$

Immediately we calculate that the two eigenvectors are orthogonal and, furthermore, observe that the matrix is singular (one of the eigenvalues was 0). Hence this matrix is not invertable.

Diagonalizing a Matrix

Solving those equations took us away from the constructions in the figures that we had before, so we now return to that way of thinking. Eigenvalues and eigenvectors make vector operations simple, where a little thought brings up the idea of a diagonal matrix. (Can you work out the matrices that arise in the reflection example?) Most importantly – and this is a really big point for understanding linear algebra — the eigenvectors can diagonalize any (real) symmetric square $(n \times n)$ matrix A (and also some others).

To understand how this works, put the eigenvectors together as the columns to define a new matrix S (perhaps think of each eigenvector defining an independent direction):

$$AS = A \begin{bmatrix} | & | & & | \\ | & | & & | \\ X_1 & X_2 & \cdots & X_n \\ | & | & & | \end{bmatrix} = \begin{bmatrix} | & | & & | \\ | & | & & | \\ \lambda_1 X_1 & \lambda_2 X_2 & \cdots & \lambda_n X_n \\ | & | & & | \end{bmatrix} = \tag{10.10}$$

$$= \begin{bmatrix} | & | & & | \\ | & | & & | \\ X_1 & X_2 & \cdots & X_n \\ | & | & & | \end{bmatrix} \begin{bmatrix} \lambda_1 & & & \\ & \lambda_2 & & \\ & & \ddots & \\ & & & \lambda_n \end{bmatrix} \tag{10.11}$$

where the second step simply applied the eigenvector equation (Eq. 10.9).

Critical point: keep these matrices in the right order; multiplication is not commutative. Remember that AB is not necessarily the same as BA for matrices. (Heuristic: think of applying one to the vector and then the other to the resulting vector. You can remove your shoes and then your socks but not the reverse.)

$$AS = S\Lambda \tag{10.12}$$

$$S^{-1}AS = \Lambda \tag{10.13}$$

$$A = S\Lambda S^{-1} \tag{10.14}$$

The second pair of equations are especially interesting. What is A^2 equal to? A^3 equal to? (The powers only appear on the eigenvalues!) Since the determinant talks about "area transformations" it follows that the determinant of a matrix equals the product of its eigenvalues. And so on.

It's worthwhile to sketch out these equations for a REAL SYMMETRIC MATRIX $\mathcal{A}$. For all such matrices, the eigenvectors are perpendicular if the eigenvalues are different. The symmetry gives us a lot:

$$\mathcal{A} = S\Lambda S^{-1} \Rightarrow Q\Lambda Q^T$$

or, in other words,

$$Q^T Q = I$$

.

Writing this out, we get:

$$\mathcal{A} = \left[\begin{array}{ccc} | & & | \\ X_1 & | & X_2 \\ | & & | \end{array}\right] \left[\begin{array}{cc} \lambda_1 & 0 \\ 0 & \lambda_2 \end{array}\right] \left[\begin{array}{c} X_1 \\ - - - \\ X_2 \end{array}\right] \tag{10.15}$$

or

$$\mathcal{A} = \lambda_1 \vec{X_1} \vec{X_1}^T + \lambda_1 \vec{X_2} \vec{X_2}^T +$$

This is an expansion of $\mathcal{A}$ into a sum of RANK 1 MATRICES, or, in other words, the same kinds of expansions we have been developing over several lectures! Notice in particular that each of the products on the right side is a column vector times a row vector (an 'outer' product).

Where we're going (in case you haven't guessed) is that these projections into the different eigenvector spaces are the projections into the principle components. There's one more thing to do before we get there.

10.2.3 SVD

The huge generalization of diagonalization to non-square matrices is called SINGULAR VALUE DECOMPOSITION, what Strang (p. 363) calls a "highlight of linear algebra." Now we get one set of vectors for the "row space" and one set of vectors for the "column space." These are called singular vectors now (instead of eigenvectors) but they're related. Let A now be an $m \times n$ matrix. We can get a square matrix by forming the product between A and its transpose. There are two ways to do this: multiply AA^T or multiply $A^T A$. Importantly, $AA^T \neq A^T A$ (think about the dimensions of both of these products); nor is their eigenstructure is not the same. But they are not unrelated and something really interesting emerges.

Denote the eigenvectors of AA^T by $\vec{u}$ and the eigenvectors of $A^T A$ by $\vec{v}$. We're going to be exploiting the fact that $A(A^T A) = (AA^T)A$; We then can write the system of equations:

$$
\begin{aligned}
A\vec{v_1} &= \sigma_1 \vec{u_1} && (10.16)\\
A\vec{v_2} &= \sigma_2 \vec{u_2} && (10.17)\\
&\vdots && (10.18)\\
A\vec{v_r} &= \sigma_r \vec{u_r} && (10.19)\\
& && (10.20)
\end{aligned}
$$

A calculation reveals the connection. Start with

$$A^T A\vec{v_1} = \lambda_1 \vec{v_1} \tag{10.21}$$

and multiply both sides by A (on the left):

$$AA^T \underbrace{A\vec{v_1}}_{\vec{u_1}} = \lambda_1 \underbrace{A\vec{v_1}}_{\vec{u_1}} \tag{10.22}$$

since $AA^T u_1 = \lambda_1 u_1$.

One last point. Let the RANK (an effective measure in independence) of A be r. Because A is not square, we need (n - r) more $\vec{v_i}$ and (n - r) more $\vec{u_i}$ to square it out.[2] In vector/matrix form this becomes $AV = U\Sigma$ (where the rest of Σ is padded with 0's.)

$$A \begin{bmatrix} \vec{v_1} & \cdots & \vec{v_n} \end{bmatrix} = \begin{bmatrix} \vec{u_1} & \cdots & \vec{u_m} \end{bmatrix} \begin{bmatrix} \sigma_1 & & \\ & \ddots & \\ & & \sigma_r \end{bmatrix} \tag{10.23}$$

In size, this matrix equation is $m \times n)(n \times n) = (m \times m)(m \times n)$.

Thus we have the SINGULAR VALUE DECOMPOSITION

$$A = U\Sigma V^T. \tag{10.24}$$

By construction (remember, eigenvectors are orthogonal) the matrices U and V^T are ORTHOGONAL MATRICES; that is, the rows and columns are orthonormal vectors so that when applied to a vector they only rotate it or reflect it – they do not change its length. Furthermore, since e.g. $V^T V = VV^T = I \Rightarrow V^T = V^{-1}$. Thus we have a beautiful picture of applying a matrix: first we rotate to the right coordinates then stretch than rotate back. See Fig. 10.9.

[2] these additional vectors come from the nullspace of A and of A^T, respectively.

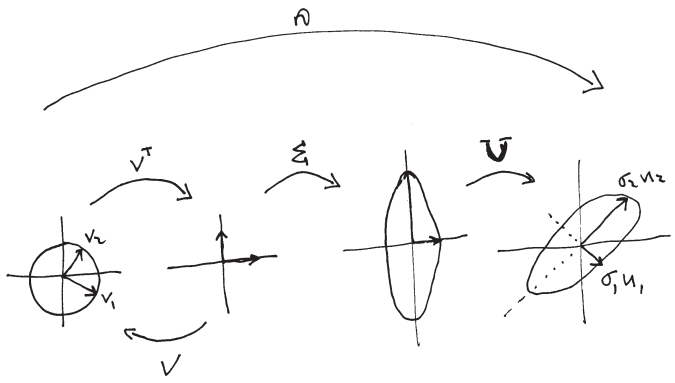

Figure 10.9: Let $A = U\Sigma V^T$. This amounts to a rotate, stretch, unrotate. Figure after Strang's fig 6.8

An example

This numerical example comes from Strang, p 366. Let

$$A = \begin{bmatrix} 2 & 2 \\ -1 & 1 \end{bmatrix}$$

so

$$A^T A = \begin{bmatrix} 5 & 3 \\ 3 & 5 \end{bmatrix}$$

which we check is symmetric by inspection. Its eigenvectors are

$$v_1 = \begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{bmatrix} \text{ and } v_2 = \begin{bmatrix} \frac{-1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{bmatrix}$$

and the eigenvalues are $\{8, 2\}$.

Now, Av_1 is in the direction of u_1:

$$Av_1 = \begin{bmatrix} 2 & 2 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{bmatrix} = \begin{bmatrix} \frac{2}{\sqrt{2}} \\ 0 \end{bmatrix}$$

which implies the unit vector $u_1 = [1\ 0]^T$. Furthermore, observing $Av_1 = 2\sqrt{2}u_1 \Rightarrow \sigma_1 = 2\sqrt{2}$ and $\sigma_1^2 = 8$.

Next,

$$Av_2 = \begin{bmatrix} 2 & 2 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} \frac{-1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{bmatrix} = \begin{bmatrix} 0 \\ \sqrt{2} \end{bmatrix}$$

which implies the unit vector $u_2 = [0\ 1]^T, \sigma_2 = \sqrt{2}$ and $\sigma_1^2 = 2$.

Putting the pieces together, we get for the SVD,

$$A = U\Sigma V^T \Rightarrow \begin{bmatrix} 2 & 2 \\ -1 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 2\sqrt{2} & 0 \\ 0 & \sqrt{2} \end{bmatrix} \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{-1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{bmatrix}.$$

10.2.4 Principle Components (finally)

Now we return to the problem of finding the best line; i.e. maximizing eq. 10.2.1. With what we've now learned, we have: With $\vec{u}_1, \ldots, \vec{u}_p$ as the columns of U (rows of U^T),

$$\arg\max_{\vec{w}, \|\vec{w}\|=1} \left\| X^\mathsf{T} \vec{w} \right\| = \arg\max_{\vec{w}, \|\vec{w}\|=1} \left\| V\Sigma U^\mathsf{T} \vec{w} \right\| = \vec{u}_1.$$

Remember that U and V are orthogonal matrices, so they simply rotate vectors without scaling them.

This shows that $\vec{u}_1$ is a direction vector that represents the line closest to all of the observations (in a least squares sense) in X—projection onto $\vec{u}_1$ provides the

"best" linear, one-dimensional representation of the data in X. What if we want a two dimensional representation? We don't need to worry about the component of the data captured by $\vec{u}_1$ anymore, so we need to find a line that is closest to the *residuals* of our data points, after subtracting away the component of the data captured by $\vec{u}_1$. The residual for an observation $\vec{x}_i$ is given by $\vec{x}_i - (\vec{x}_i \cdot \vec{u}_1)\vec{u}_1$ (subtracting away the component of the observation lying along $\vec{u}_1$), and we can see that this is orthogonal to $\vec{u}_1$ (since $\vec{u}_1$ is unit length). So we simply need to find the line closest to our data points (in a least squares sense) that is orthogonal to $\vec{u}_1$. An analysis similar to the above shows that this is $\vec{u}_2$. Together, projecting onto $\vec{u}_1$ and $\vec{u}_2$ provides the best two-dimensional representation of our data. We can continue this process for all p dimensions of our data until we represent it perfectly.

The left singular vectors of X ($\vec{u}_1, \ldots, \vec{u}_p$) are called the *principal components* of X, and act as a basis for our observations. We can express our observations in this basis by projecting each observation vector onto each principal component vector, computing $\Theta = X^\mathsf{T} U$—the columns of Θ are n-element vectors providing coordinates for each observation in our new basis of principal components. By construction, picking only the d principal components associated with the largest singular values minimizes the mean squared error of our representation for all choices of d basis vectors (i.e. it is the "best" d-dimensional representation of the data that can be achieved by trying to represent the data using only d vectors).

10.3 Eigenfaces

Finally we can understand eigenfaces. Build a big data matrix, X, where each face image is a column. Remember to subtract the mean to center the data. Calculate the left singular vectors – this is the eigenface basis. Note that, unlike the standard basis that we started with this one has lots of contributions from the different pixels in each "direction." Fig. 11.5.

A trick that's often used is the same as the intermediate step in the example: calculate the v's first, then form Av to get the u's. More on this in the next lecture.

For recognition, simply project a new face image into this basis and find the cluster (or image) to which it is closest. Working in the first k coordinates makes the entire process more efficent. Since these capture a lot of (most of) the structure, then this should help a lot.

It was a great hope of the facial recognition community that the main structure of the face would be captured in the first few dimensions, and other (annoying) complications – such as lighting effects – would be relegated to dimensions that could be ignored. Have a quick look at the two images in Fig. 10.11 – are they the same? – to realize how difficult the problem is. Even though all of the scale and pose issues have been dealt with by the manner in which the pictures were taken, lighting changes remain non-trivial. In situations where one can control scale, pose, lighting, and so on template matching schemes can work. They are widely deployed in checking for

Figure 10.10: Some Turk eigenfaces (finally). How many should there be? How would you use these for image reconstruction? Note that this connects to the concept of IMAGE COMPRESSION which we won't have time to discuss.

driver's license fraud (search the web for Face Explorer, for example). This reveals an interesting dichotomy: understanding general recognition systems may well be a different problem from engineering systems for specific tasks. The images on driver's licences are highly stereotyped; images in the wild much less so.

But remember, the images are not perfect. There's one more possibility: noise. So a statistical approach may also be warrented. We turn to that next.

10.4 Notes

The linear algebra comes from G. Strang, Introduction to Linear Algebra.

D. Holtmann-Rice helped enormously in organizing the material for this chapter.

Eigenfaces: M. Turk and A. Pentland, Eigenfaces for Recognition, Journal of Cognitive Neuroscience, 3(1), 1991; see also the Scholarpedia page.

There's an interesting literature on the web about the Dept. of Homeland Security and biometric markers to establish identity. Face recognition is one area in which template matching is applied.

A popular tutorial on PCA among neuroscience instructors is "A Tutorial on Principal Component Analysis" by Jonathon Shlens; you can grab it off the web.

Figure 10.11: Look at the two images in this figure quickly: are they images of the same person? What's changed? Answer: yes; and the lighting has changed. Complications such as this suggest that working in raw pixel spaces is limited to certain technical conditions. Figure from Belhumeur-Kriegman.

Chapter 11

Statistical View of Principle Components

11.1 Introduction

PCA also has an important statistical interpretation, which may be even more common in your experience. Perhaps you've heard that the principle components "maximize variance." We're going to have to introduce a few basic notions, before explaining what this means.

11.2 Motivating Material

Before doing PCA, we describe a number of constructions to "remind" the reader of basic ideas. We start off with some direct calculations to show why the concept of "sum of squares" is so fundamental in statistics.

11.2.1 Sample Mean, Sample Variance

Roles for the mean and variance – the "center" and the "spread" – of a set of data were introduced at the end of the last lecture. The mean is totally intuitive: the center treats each data point homogeneously. Of course there is still significant possibility for variation, by taking not the data points $\{x_i\}$ but some function f of them:

$$nf(m) = \sum_{i=1}^{n} f(x_i) \tag{11.1}$$

where m denotes the mean. The simplest possible function f is the identity, so we have the formula: $m = \frac{1}{n}\sum_1^n x_i$.

Variance and standard deviation are perhaps less intuitive. One is tempted to take the average across all data points of their deviation from the sample mean, but this

just gives 0. (Working this out gives you another handle on linearity.) So instead of starting with a definition of the mean, let's start with an idea for standard deviation – a model for the spread in the data – and see what this implies about the mean.

Building on the above, let's define the spread using the absolute value of the deviation from the "mean." (Since we are seeking to calculate what this new "mean" could be, let's denote it by λ. A natural notion for the spread is:

$$s_1 = \frac{1}{n}\sum |x_i - \lambda|. \tag{11.2}$$

The choice of the mean, λ, is that value of of λ which minimizes the spread for some set of data points. Minimization via calculus runs into problems because of the absolute value function; instead we illustrate a procedure in Fig. 12.2 that lets us solve the problem algorithmically. Following this procedure reveals that the s_1 definition of spread, the absolute deviation, is connected to the median of the data.

Now let's try another measure of the spread exploiting the square rather than absolute value. To keep the units the same as distance we need to introduce the square root.

$$s_2 = \sqrt{\frac{1}{n}\sum (x_1 - \lambda)^2} \tag{11.3}$$

We can now use calculus to minimize this function to find λ. For simplicity we minimize s_2^2 since this doesn't change anything.

$$\frac{d}{d\lambda} s_2^2(\lambda) = \frac{-2\sum (x_i - \lambda)}{n} = 0 \tag{11.4}$$

which implies $\lambda = \frac{1}{n}\sum x_i$ or the classical sample average (or arithmetic mean). This is where the phrase STANDARD DEVIATION arises, because it is so naturally associated with the standard mean.

We say the mean is UNBIASED when

$$\lim_{n\to\infty} \sum_1^n (x_i - \mu) = 0 \tag{11.5}$$

for all data x_i that we may choose. (similar results hold for other unbiased estimators.)

11.2.2 The Gaussian Distribution

Because the covariance matrix arises above, let's think about what this means from a Gaussian perspective. Recall that a random variable drawn from a Gaussian p.d.f. is

$$p(X) = \frac{1}{(2\pi)^{\frac{1}{2}}\sigma} \exp\{-(x-\mu)^2/2\sigma^2\} \tag{11.6}$$

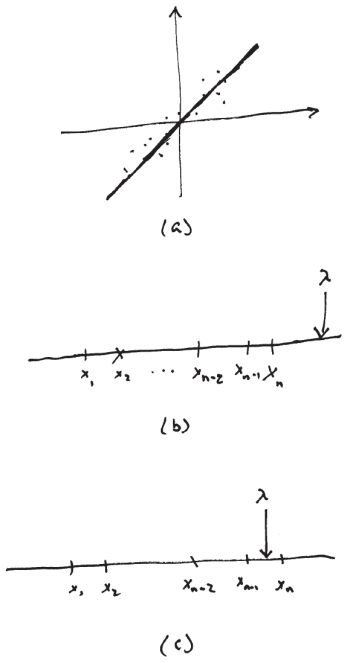

Figure 11.1: Illustrating the connection between the mean and the standard deviation. A A set of data points projected onto a line. We wish to understand how the mean of these data are related to the absolute deviation, s_1. B After projection, suppose the data points align as shown. Start with a guess for λ at the far right of the data points. C As λ is moved leftward, it passes the last data point x_n.Once this happens, notice that all further leftward movement from x_n is exactly cancelled by the approach to the first data point, x_1. After x_{n-2} is passed, it cancels with x_2, and so on until the median data point is reached (for an odd number of data points). Follows Koks p 84

where the mean is μ and the standard deviation is σ. You've all seen this before – it has the properties that you know and love.

It's worthwhile to write out the Gaussian for a 2-dimensional random variable

$$X = \begin{bmatrix} x \\ y \end{bmatrix}$$

where each of the components, x and y is a 1-dimensional Gaussian r.v. Defining the MEAN VECTOR as

$$\mu = \begin{pmatrix} \mu_x \\ \mu_y \end{pmatrix}$$

and the COVARIANCE MATRIX as

$$\Sigma = \begin{pmatrix} \sigma_x^2 & \sigma_{xy} \\ \sigma_{xy} & \sigma_y^2 \end{pmatrix}$$

we obtain the 2D normal:

$$P(x) = \frac{1}{2\pi||\Sigma||^{\frac{1}{2}}} \exp\left(-\frac{1}{2}(X-\mu)^T\Sigma^{-1}(X-\mu)\right) \tag{11.7}$$

It's instructive to compare eq. 11.6 with eq. 11.7. First, look at the exponent. scale factor to make it integrate to 1.

For D-dimensional random variables we get the multivariate Gaussian (multivariate normal) $p(X) = p(x_1, x_2, \ldots, x_D$

$$p(X) = \frac{1}{(2\pi)^{\frac{D}{2}}||\Sigma||^{1/2}} \exp\{-\frac{1}{2}(X-\vec{\mu})^T\Sigma^{-1}(X-\vec{\mu})\} \tag{11.8}$$

where now we have the vector of means and the covariance matrix Σ – how each pair of variables varies in expectation. In symbols, the i-th variable x_i differs from the i-th mean by $e_i = x_i - \mu_i$. The covariance

$$\sigma_{ij} = \sigma_{ji} = E[e_i e_j] = \sum\sum p_{ij} e_i e_j. \tag{11.9}$$

For the multidimensional Gaussian, we could apply the above linear algebra to the exponent: write $\Sigma = Q\Lambda Q^T$ since it's symmetric, so we introduce new random variables $Y = Q^{-1}X$. Then the multidimensional Gaussian factors into a product on 1-dimensional Gaussians (an amazing property!) Fig. 16.4.

$$\exp -\frac{1}{2}X^T\Sigma^{-1}X = \exp\{-\frac{1}{2}Y^T\Lambda^{-1}Y\} = \tag{11.10}$$

$$= (\exp\{-y_1^2/2\lambda_1\})(\exp\{-y_2^2/2\lambda_2\})\cdots(\exp\{-y_D^2/2\lambda_D\}) \tag{11.11}$$

and each of these λ is the original variance σ_i^2.

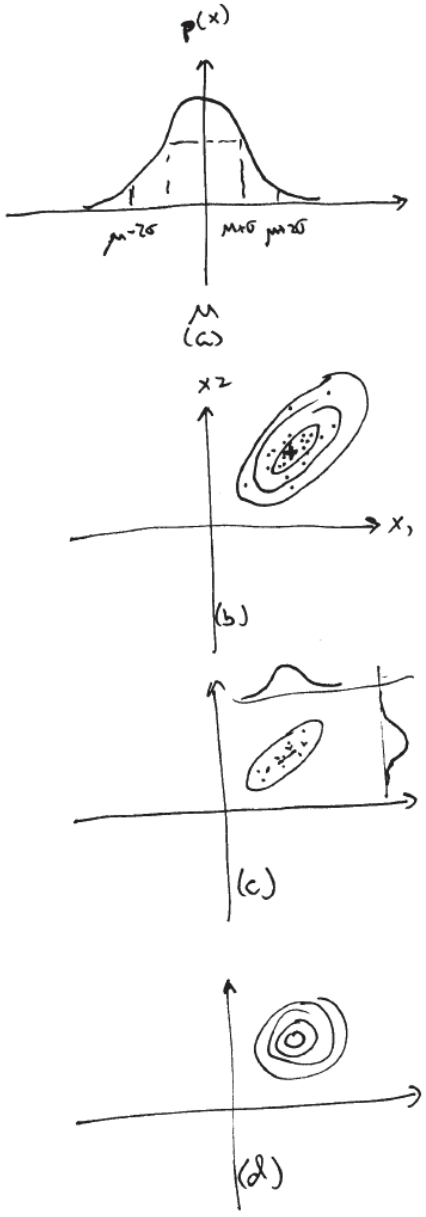

Figure 11.2: The Gaussian distribution. (a) In 1-D about 95 % of the probability density concentrates within $\pm 2\sigma$. (b) Samples from the multidimensional Gaussian collect within ellipses centered around the mean +. (c) Projections onto different lines are 1-D Gaussians. (d) A WHITENING transformation converts the covariance matrix Σ into the identity matrix I.

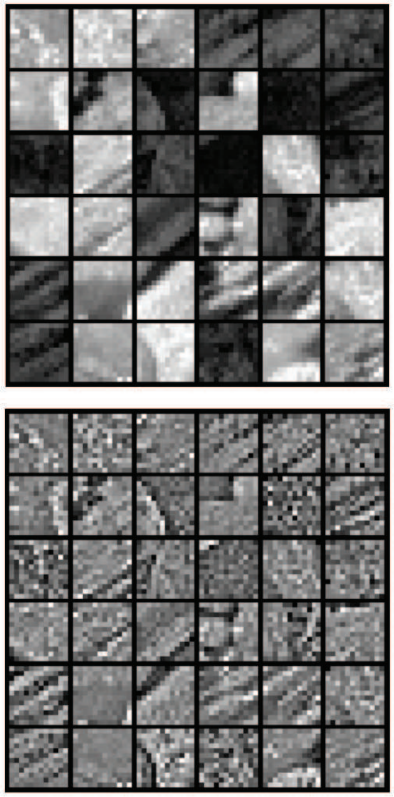

Figure 11.3: Illustration of whitening. (a) Original (12 x 12) image patches. (b) ZCA whitened patches. These should have a diagonal covariance matrix Notice how the huge correlation structure in the gray levels is decorrelated, but also how some of the "edge" structure is maintained. Figs from deeplearning wiki.

Central Limit Theorem

Whitening Data

Mahalanobis Distance

Only need mean and variance

11.3 PCA and Statistics

What do the singular values $\sigma_1, \ldots, \sigma_p$ represent? Let's take a closer look at Θ, the projection of our observations onto the principal components. Using the SVD of X once more, we can see $\Theta = X^\mathsf{T}U = V\Sigma U^\mathsf{T}U = V\Sigma$. Thus the lengths of the vectors that form the columns of Θ are given by the singular values in Σ (since all the columns of V are unit length, and multiplying a matrix by a diagonal matrix from the right simply scales its columns). So, for $\vec{u}_i$ we have $\sigma_i = \left\|X^\mathsf{T}\vec{u}_i\right\| =$

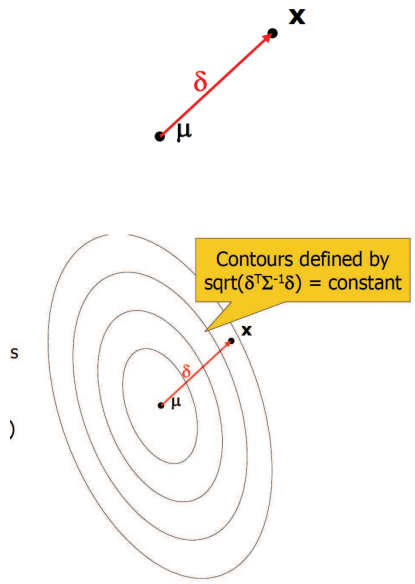

Figure 11.4: The exponent in the multivariate Gaussian can be interpreted as as a distance, the Mahalanobis distance Note: non-euclidean because it matters which direction we 'walk' in. Figs from Moore.

$\sqrt{(\vec{x}_i \cdot \vec{u}_i)^2 + (\vec{x}_2 \cdot \vec{u}_i)^2 + \ldots + (\vec{x}_n \cdot \vec{u}_i)^2}$. Now, assuming we've centered our observations (so that the mean along the rows of X is 0), this is proportional to the standard deviation of the projections of our observations onto $\vec{u}_i$. If our observations are drawn from some larger population of image patches, stocks, etc. (as is generally the case), then $\sigma_i^2/(n-1)$ gives a good estimate of the variance of the projections of samples from the population onto $\vec{u}_i$.

In fact, we can develop a statistical interpretation for the entire PCA procedure. We can construct from X a covariance matrix $C = \frac{1}{(n-1)} XX^\mathsf{T}$. Thinking of our observations as samples from some multivariate distribution (with mean 0), for $i \neq j$, C_{ij} is an estimate of the covariance between dimension i and dimension j of our data, and C_{ii} is an estimate of the variance of dimension i. Instead of taking the SVD of X^T and then considering its left singular vectors, we can work with the eigenvalues and eigenvectors of C. Since $C = \frac{1}{(n-1)} XX^\mathsf{T} = \frac{1}{(n-1)} U\Sigma V^\mathsf{T} V\Sigma U^\mathsf{T} = \frac{1}{(n-1)} U\Sigma^2 U^\mathsf{T}$, we can see C has as its eigenvectors the left singular vectors of X (our principal components), and as its eigenvalues our estimates of the variance of the projections of the observations onto the principal components.

C can be considered as a linear operator. Given some vector $\vec{w}$, $\vec{w}^\mathsf{T} C\vec{w}$ gives the variance of a linear combination of dimensions of our data with weights given by $\vec{w}$. Thus we can interpret the principal components of X as directions in our data maximizing the variance of the projections of our observations. The eigenvalues of C tell us how much of the variance in our data each principal component "explains". We can if we like aim for a specific threshold—say, 90% of the total variance—and pick $d < p$ such that the sum of the first d eigenvalues crosses this threshold. This is one way of determining a reduced dimensionality for our data that is (ever so slightly) less arbitrary than just trying a couple and seeing what works well.

11.4 Eigenfaces

Finally we can understand eigenfaces. Build a big data matrix, X, where each face image is a column. Remember to subtract the mean to center the data. Calculate the empirical covariance matrix XX^T. Calculate its eigenvectors. This is the eigenface basis. Note that, unlike the standard basis that we started with this one has lots of contributions from the different pixels in each "direction." Fig. 11.5.

For recognition, simply project a new face image into this basis and find the cluster (or image) to which it is closest. Working in the first k coordinates makes the entire process more efficent.

11.5 Summarizing our Perspective

We have developed a *global* view toward object recognition – well, picture recognition. Suffers from having "perfectly" scaled versions of faces (or whatever) without glasses

Figure 11.5: Some Turk eigenfaces (finally). How many should there be? How would you use these for image reconstruction? Note that this connects to the concept of IMAGE COMPRESSION which we won't have time to discuss.

etc. While this can be made to work for certain engineered problems, in general it seems unlikely: how would a brain "learn" which images are perfectly cropped, scaled versions of every possible object we care about.

And the story gets worse when we think about the real world. Most importantly, the image will change with lighting (Fig. 11.6) as well as pose and distance and ...

11.5.1 More Curiosities of Facial Perception

There's tons more to be said about face recognition, but this will have to wait. For now we close with another tantalizing illusion of face recognition: whether the orientation of the mouth region is correctly assessed as smiling or frowning. See the Thatcher illustion in Fig. 11.7.

For some nice examples on the web, see e.g. http://thatchereffect.com/

We now return to biology, and learn about a more localized – and staged – approach.

11.6 Notes

Standard deviation material follows Koks, Explorations in Mathematical Physics.

Mahalanobis distance figures from Andrew W. Moore's online tutorial "Gaussians".

whitening example from deeplearning.stanford.edu/wiki/index.php/Exercise:PCA_and_Whitenin

Figure 11.6: An image of a cropped face changes drastically with lighting. How might an algorithm correct for these lighting changes? Image from Mumford.

Figure 11.7: The Thatcher illusion. Thompson, P. (1980). Margaret Thatcher: a new illusion. Perception. doi: 10.1068/p090483