The Best Basis

In this chapter, we will consider the problem of compact data representations. In particular, we will look at how we might define a "best basis", and the result will be that the best basis is given by the **eigenvectors** of the **covariance matrix**. This well known result goes by several names, depending on the discipline. For example, it goes by the Karhunen-Loéve (KL) expansion, Principal Components Analysis (PCA), and Principal Orthogonal Decomposition (POD) to name a few. We will call it PCA as a nod to the statistics.

Perhaps suprisingly, we will compute these eigenvectors without having to actually compute the covariance matrix. Instead, the SVD will play a very large role.

0.1 Motivating Example

Suppose I have some data in $\mathbb{R}^2$, illustrated by the data plot in Figure 1. We see that the data looks "mostly" one dimensional in that the data is basically all very close to some line. In that case, we should be able to store it as constant multiples of a single vector, which takes the data from two dimensional to one dimensional.

We might also notice that the line does not go through the origin, so that the data, as represented here, is not actually lying in a subspace- We'll need to fix that first so that we can use our linear algebra.

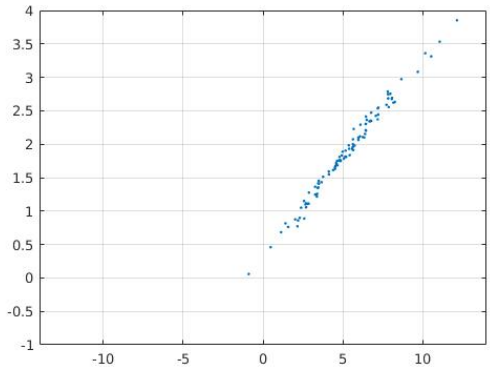

Figure 1: Data in the plane. It looks like it is close to being one-dimensional.

In what follows, we'll mean-subtract the data (the mean being a vector in $\mathbb{R}^2$), so that the data is centered at the origin. Then we'll need to find a way of constructing the best vector for the data.

0.2 Introductory Computations

Suppose we have a set of p data points in $\mathbb{R}^n$,

$$\left\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(p)}\right\}$$

Furthermore, let X be the $n \times p$ matrix formed by storing the data as columns:

$$X = [\boldsymbol{x}^{(1)} \ \ldots \ \boldsymbol{x}^{(p)}]$$

As we mentioned, we'll assume that the mean vector for the data (the mean being a vector in $\mathbb{R}^n$) is zero. With that, we will assume that the vector contained in some subspace spanned by some set of vectors.

Now, given any basis of $\mathbb{R}^n$, say n linearly independent vectors $\boldsymbol{\phi}_i$ each in $\mathbb{R}^n$ and stored as the columns of matrix Φ:

$$\Phi = [\boldsymbol{\phi}_1 \ \boldsymbol{\phi}_2 \ \ldots \boldsymbol{\phi}_n],$$

we can write any vector $\mathbf{x} \in \mathbb{R}^n$ in terms of this basis:

$$\boldsymbol{x} = \alpha_1\boldsymbol{\phi}_1 + \alpha_2\boldsymbol{\phi}_2 + \cdots + \alpha_n\boldsymbol{\phi}_n = \sum_{k=1}^{n} \alpha_k\boldsymbol{\phi}_k = \Phi\boldsymbol{\alpha} \tag{1}$$

We will assume our basis is orthonormal, so that the coefficients α_k are easy to compute. They are simply the dot product between $\mathbf{x}$ and the k^{th} basis vector, and that leads us to a nice matrix representation for the vector of coordinates:

$$\alpha_k = \boldsymbol{\phi}_k^T\boldsymbol{x} \quad \Rightarrow \quad \boldsymbol{\alpha} = \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_n \end{bmatrix} = \begin{bmatrix} \boldsymbol{\phi}_1^T\boldsymbol{x} \\ \boldsymbol{\phi}_2^T\boldsymbol{x} \\ \vdots \\ \boldsymbol{\phi}_n^T\boldsymbol{x} \end{bmatrix} = \begin{bmatrix} \boldsymbol{\phi}_1^T \\ \boldsymbol{\phi}_2^T \\ \vdots \\ \boldsymbol{\phi}_n^T \end{bmatrix} \boldsymbol{x} = \Phi^T\boldsymbol{x}$$

Notice that if we now substitute this expression in for $\boldsymbol{\alpha}$ in Equation 1, we get:

$$\boldsymbol{x} = \Phi\boldsymbol{\alpha} = \Phi\Phi^T\boldsymbol{x}$$

Recall that if Φ is a square matrix (with orthonormal columns), then $\Phi\Phi^T$ is the identity, but if it is not, then this represents a projection of $\mathbf{x}$ into the subspace spanned by the columns of Φ.

Introduction of Error

Let's consider this second choice- Suppose we are interested in the subspace spanned by the first k columns of Φ (where $k << n$). Then we can write any point $\mathbf{x}$ in the usual way, but split the expression after the k^{th} column:

$$\boldsymbol{x} = \sum_{j=1}^{k} \alpha_j\boldsymbol{\phi}_j + \sum_{j=k+1}^{n} \alpha_j\boldsymbol{\phi}_j \doteq \boldsymbol{x}_{\text{in}} + \boldsymbol{x}_{err}$$

The first term is what we're calling the part of $\boldsymbol{x}$ that we're keeping. The second term of the sum is our **error** in using k basis vectors to express $\boldsymbol{x}$:

$$\boldsymbol{x}_{\text{err}} = \sum_{j=k+1}^{n} \alpha_j\boldsymbol{\phi}_j$$

And, by assuming that the vectors $\boldsymbol{\phi}_i$ are orthonormal, we can write the magnitude of the error vector for a single point $\mathbf{x}$ as the following (this was proved earlier using the Pythagorean Theorem):

$$\|\boldsymbol{x}_{\text{err}}\|^2 = \alpha_{k+1}^2 + \cdots + \alpha_n^2$$

Before we continue, if the basis is orthonormal, then we could write, by the Pythagorean Theorem, that

$$\|\boldsymbol{x}\|^2 = \|\boldsymbol{x}_{\text{in}}\|^2 + \|\boldsymbol{x}_{\text{err}}\|^2$$

The quantity on the left side of the equation, $\|\boldsymbol{x}\|^2$, is constant. Therefore, on the right side of the equation, by choosing different basis vectors $\boldsymbol{\phi}_i$, *minimizing the error* is the same as *maximizing the first term.*

Now we're ready to proceed with determining the best basis.

The Best "Basis" for a set of Data

If we think about what characterizes a good basis for a given set of data, we might come up with the following list:

1. Since the best basis will naturally have its origin at the centroid of the data, **we will assume that the data has been mean subtracted**.

2. The best basis should be orthonormal.

3. The data does not take up all of $\mathbb{R}^n$- It lies in some subspace.

4. If we have p data points, the error using k columns will be defined by the mean squared error:

$$\text{Error} = \frac{1}{p-1}\sum_{j=1}^{p} \|\boldsymbol{x}_{\text{err}}^{(j)}\|^2 \tag{2}$$

   Note that the scaling factor $1/(p-1)$ is there for convenience when we bring in the covariance matrix. However, the location of the minimizer of the error is not affected by the scaling factor, and could be just left off.

Our goal is to choose Φ that will minimize the given error function in Equation 2. The following exercises outline that process.

Exercises: Bringing in the Covariance

1. Show that the j^{th} coordinate for a single data point $\mathbf{x}$ can be written as:

$$\alpha_j^2 = \boldsymbol{\phi}_j^T \boldsymbol{x}\boldsymbol{x}^T \boldsymbol{\phi}_j$$

2. Show that the inner product and sum can be switched. That is, for a single vector $\boldsymbol{\phi}$, show that we can write:

$$\sum_{i=1}^{p} \boldsymbol{\phi}^T \boldsymbol{x}^{(i)} \left(\boldsymbol{x}^{(i)}\right)^T \boldsymbol{\phi} = \boldsymbol{\phi}^T \left[\sum_{i=1}^{p} \boldsymbol{x}^{(i)} \left(\boldsymbol{x}^{(i)}\right)^T\right] \boldsymbol{\phi}$$

3. (Remark) We can write the covariance matrix of X ($n \times p$ and mean subtracted) in the following way:

$$C = \frac{1}{p-1} XX^T = \frac{1}{p-1}\sum_{i=1}^{p} \boldsymbol{x}^{(i)} \left(\boldsymbol{x}^{(i)}\right)^T$$

4. Here is a useful lemma that we'll use later. Suppose $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n$ are fixed values, and further, suppose we can choose n real numbers, $p_1, p_2, \cdots, p_n$ which are all non-negative and sum to one (like a probability measure). Then the maximum and minimum of the convex combination below is given by:

$$\lambda_n \leq p_1\lambda_1 + p_2\lambda_2 + \cdots + p_n\lambda_n \leq \lambda_1$$

   How is that? The minimum is found by making p_n as large as possible, and the rest as small as possible- $(0, 0, \cdots, 0, 1)$, which gives λ_n. Similarly, the maximum is found by taking p_1 as large as possible, and making the rest as small as possible. This returns the value λ_1.

0.3 The Best Basis

We're now just about ready to find the best basis. First, let's summarize what we have shown in the previous exercises:

$$\begin{aligned}
\frac{1}{p-1}\sum_{i=1}^{p}\|\boldsymbol{x}_{\text{err}}^{(i)}\|^2 &= \frac{1}{p-1}\sum_{i=1}^{p}\left(\sum_{j=k+1}^{n}\left(\alpha_j^{(i)}\right)^2\right) \\
&= \frac{1}{p-1}\sum_{i=1}^{p}\left(\sum_{j=k+1}^{n}\boldsymbol{\phi}_j^T\boldsymbol{x}^{(i)}\left(\boldsymbol{x}^{(i)}\right)^T\boldsymbol{\phi}_j\right) \\
&= \sum_{j=k+1}^{n}\left(\frac{1}{p-1}\sum_{i=1}^{p}\boldsymbol{\phi}_j^T\boldsymbol{x}^{(i)}\left(\boldsymbol{x}^{(i)}\right)^T\boldsymbol{\phi}_j\right) \\
&= \sum_{j=k+1}^{n}\boldsymbol{\phi}_j^T\left[\frac{1}{p-1}\sum_{i=1}^{p}\boldsymbol{x}^{(i)}\left(\boldsymbol{x}^{(i)}\right)^T\right]\boldsymbol{\phi}_j \\
&= \sum_{j=k+1}^{n}\boldsymbol{\phi}_j^T C\boldsymbol{\phi}_j
\end{aligned}$$

Conclusion so far: To minimize the mean-squared error on the $k-$term expansion $\{\boldsymbol{\phi}_i\}_{i=1}^{k}$, we need to minimize:

$$\sum_{j=k+1}^{n}\boldsymbol{\phi}_j^T C\boldsymbol{\phi}_j$$

with the constraints that the vectors $\left\{\boldsymbol{\phi}_j\right\}_{j=1}^{n}$ form an orthonormal set (you may have noted that the value of k has been left undecided). From our previous note, we should also recall that minimizing the given error term is the same thing as maximizing the term:

$$\sum_{j=1}^{k}\boldsymbol{\phi}_j^T C\boldsymbol{\phi}_j$$

Notice that the best $k-$dimensional basis may not be unique, but any optimal choice of basis should give you the same $k-$dimensional subspace. Later, we'll look at how we might do this, but there is an easier way first.

Our method will be to optimize for $k = 1$ first, then we'll look for the next best vector, $k = 2$, then the next best, $k = 3$, and so on. Now, let $k = 1$, and we'll maximize the appropriate quantity (remember that the unknowns are in the vector $\boldsymbol{\phi}_1$!)

$$\max_{\phi_1\neq 0}\frac{\boldsymbol{\phi}_1^T C\boldsymbol{\phi}_1}{\boldsymbol{\phi}_1^T\boldsymbol{\phi}_1}$$

Theorem: The maximum (over non-zero ϕ) of the quantity

$$\frac{\phi^T C\phi}{\phi^T\phi}$$

where C is the covariance matrix of a data set X occurs when $\boldsymbol{\phi} = \boldsymbol{v}_1$, the first eigenvector of C, where λ_1 is the largest eigenvalue of C.

The proof will proceed as several exercises.

Exercises: The Best Basis

1. Justify the following:

   Let $\{\lambda_i\}_{i=1}^n$, $\{\boldsymbol{v}_i\}_{i=1}^n$, be the eigenvalues and eigenvectors (respectively) of the covariance matrix C, where
   $$\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_n$$
   Then we can write C as:
   $$C = \lambda_1 \boldsymbol{v}_1 \boldsymbol{v}_1^T + \ldots + \lambda_n \boldsymbol{v}_n \boldsymbol{v}_n^T = V \Lambda V^T$$
   with $V^T V = V V^T = I$.

2. We can write any vector $\boldsymbol{\phi}$ as a linear combination of the eigenvectors:
   $$\boldsymbol{\phi} = a_1 \boldsymbol{v}_1 + \ldots + a_n \boldsymbol{v}_n = V \boldsymbol{a}$$

3. From the previous exercise, show that:
   $$\boldsymbol{\phi}^T \boldsymbol{\phi} = a_1^2 + a_2^2 + \ldots + a_n^2$$

4. Show that:
   $$\boldsymbol{\phi}^T C \boldsymbol{\phi} = \lambda_1 a_1^2 + \lambda_2 a_2^2 + \ldots + \lambda_n a_n^2$$

5. Justify the following: For any nonzero vector $\boldsymbol{\phi}$,
   $$\frac{\boldsymbol{\phi}^T C \boldsymbol{\phi}}{\boldsymbol{\phi}^T \boldsymbol{\phi}} = \frac{\lambda_1 a_1^2 + \lambda_2 a_2^2 + \ldots + \lambda_n a_n^2}{a_1^2 + c_2^2 + \ldots + a_n^2} \leq \lambda_1$$
   with equality if and only if $\boldsymbol{\phi} = \boldsymbol{v}_1$.

   By the way, this works in both directions:
   $$\lambda_n = \lambda_{\min} \leq \frac{\boldsymbol{\phi}^T C \boldsymbol{\phi}}{\boldsymbol{\phi}^T \boldsymbol{\phi}} =\leq \lambda_{\max} = \lambda_1$$

So far, we have shown how to find the best one dimensional basis vector! To find the next best basis vector, we need to determine a vector $\boldsymbol{\phi}$ that satisfies the following:

$$\max_{\boldsymbol{\phi} \perp \boldsymbol{v}_1} \frac{\boldsymbol{\phi}^T C \boldsymbol{\phi}}{\boldsymbol{\phi}^T \boldsymbol{\phi}}$$

which, by the same argument as in the previous theorem, is going to be the second eigenvector of C- this is a consequence of the Spectral Theorem, which says that our eigenvectors are orthogonal.

The Best Basis Theorem

Suppose that:

- X is an $n \times p$ matrix of p points in $\mathbb{R}^n$ with mean $\bar{x} \in \mathbb{R}^n$.
- X_m is the $n \times p$ matrix of mean-subtracted data.
- C is the covariance of X, $C = \frac{1}{p-1} X_m X_m^T$

Then the best (in terms of the mean-squared error defined earlier) orthonormal $k-$dimensional basis is given by the leading k eigenvectors of C, for any k.

0.3.1 Another View of the Best Basis

Consider p points in $\mathbb{R}^n$ as before:

$$\left\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(p)}\right\}$$

and suppose we project this data onto some unit vector $\boldsymbol{v}$. The result would be p points along the (unit) vector $\mathbf{v}$. We want to consider the mean and variance of this data as one-dimensional data:

$$\left\{\boldsymbol{v}^T\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{v}^T\boldsymbol{x}^{(p)}\right\}$$

Some notes about this list of data:

- Assuming that the data has zero mean (if not, mean-subtract the data), we showed earlier that the variance of the projected data is given by:

$$\mathbf{v}^T C\mathbf{v} = \frac{\mathbf{v}^T C\mathbf{v}}{\mathbf{v}^T\mathbf{v}}$$

- In the particular case where $\mathbf{v}$ is actually the first eigenvector of C, what does this expression simplify to?

Therefore, the subspace we compute is the one which is "capturing" the most variance of the data.

0.4 Connections to the SVD

Consider the SVD of the $n \times p$ matrix X, with rank k:

$$X = U\Sigma V^T$$

so that U is $n \times k$ (and has orthonormal columns), Σ is $k \times k$ diagonal matrix with non-zero entries along the diagonal, and V is $p \times k$ with o.n. columns.

By formulating the covariance, we see that:

$$C = \frac{1}{p-1}XX^T = \frac{1}{p-1}U\Sigma V^T V\Sigma^T U^T = U\left(\frac{1}{p-1}\Sigma^2\right)U^T$$

And this is the eigenvector-eigenvalue decomposition of C. Comparing this to the Best Basis Theorem, we see that:

> The best basis vectors for the column space of X are the first k columns of U. Similarly, the best basis vectors for the row space of X are the first k columns of V.

Furthermore, the SVD gives a beautiful symmetry to the problem: That is, knowledge of the best basis vectors of the rowspace gives the best basis vectors of the columnspace (and vice-versa) by the relationships:

$$X\boldsymbol{v}_i = \sigma_i\boldsymbol{u}_i \qquad X^T\boldsymbol{u}_i = \sigma_i\boldsymbol{v}_i$$

where σ_i can be computed from the eigenvalues of the covariance:

$$\lambda_i = \frac{\sigma_i^2}{p-1}$$

This author has seen this symmetry relegated to being a "trick" in at least one engineering paper!

0.5 Computation of the Rank

Analytically, we can determine the rank of a matrix by counting how many non-zero singular values are in the SVD. Unfortunately, numerically, this can be difficult to do- Instead of being exactly zero, some of the eigenvalues may just be very small (like 10^{-12}). Another issue may come up with noisy data- no $k-$dimensional subspace actually contains *all* of the data. We consider ways of dealing with this next.

Numerically there are a couple of ways of *estimating* a good value of the rank k:

- If there is a large gap in the graph of the eigenvalues, we'll use that as our rank (the index of the eigenvalue just prior to the gap).

  As a numerical example, suppose the eigenvalues are:

  $$\{345, \quad 62, \quad 8, \quad 0.1, \quad 0.001, \quad 0.00001\}$$

  In this case, we might have reason to believe that the data can be safely truncated using three dimensions.

- The unscaled eigenvalues are hard to compare- we've seen that they might be scaled by p or n, and the exact value is not as important as their relative sizes. We should always consider the normalized eigenvalues- let $\tilde{\lambda}_i$ be the unscaled eigenvalues of C. Then:

  $$\lambda_i = \frac{\tilde{\lambda}_i}{\sum_{j=1}^{n} \tilde{\lambda}_j}$$

  are the normalized eigenvalues of C. Note that they are all positive, and sum to one.

  As an example, using the previous set of eigenvalues, the normalized values would be:

  $$\left\{0.831, \quad 0.149, \quad 0.019, \quad 0.0002, \quad 2.4 \times 10^{-6}, \quad 2.4 \times 10^{-8}\right\}$$

  Looking at these numbers, we might change our mind from our previous answer. Since the sum of the first two values is 0.9805, we might just keep two dimensions.

- The normalized eigenvalue λ_i can be interpreted as the percentage of the total variance (or total energy) captured by the i^{th} eigenspace (the span of i^{th} eigenvector).

  In our previous example, that means that the subspace spanned by the first eigenvector "explains" or "encapsulates" about 83% of the overall variance, and the two dimensional subspace spanned by the first two eigevectors encapsulates about 98% of the overall variance in the data.

- We can think of the dimension as a function of the total energy (or total variance) that we want to encapsulate in a $k-$dimensional subspace.

- Define the principal components dimension as the number of eigenvectors required to encapsulate a given percentage of the total variance of the data:

  $$\text{PC}_d = k$$

  where k is the smallest integer so that:

  $$\lambda_1 + \ldots + \lambda_k \geq d$$

  (Recall that these are the normalized, ordered eigenvalues!)

- The value you choose for PC_d will be problem dependent. For example, if your data is the graph of a smooth function with no noise, you might choose $PC_d = 0.99$. If there is a lot of noise corrupting the data, you might choose $PC_d = 0.6$.

0.6 Matlab and PCA

As we have seen in the previous sections, there are many ways that we might compute the principal components. We'll look at several, and discuss the pros and cons.

Using the SVD directly

In this case, if X has been previously loaded, and we assume our data is organized column-wise (X is $n \times p$), then we could use the following to compute the principal components:

```
[n,p]=size(X);
mx=mean(X,2);
Xm=X-repmat(mx,1,p);
[U,S,V]=svd(Xm,0);
ss=diag(S).^2    % Eigenvalues of the covariance (scaled)
ss=ss./sum(ss);  % Normalized eigenvalues.
```

Some comments on this code:

- It is critical to mean subtract your data! Otherwise, your first eigenvector points to the mean, which is a wasted dimension.
- The additional argument (0) in the SVD command can be used when p is much larger than n. Matlab will only return the first n columns of V.
- The basis for the data is taken from the first k columns of U, where k is the dimension you want.
- To visualize the singular values, plot them- it's convenient to also plot the point values: `plot(ss,'k-*');`
- You can use `cumsum(ss)` to look at the cumulative sums of the normalized singular values to see how many you want to use.

This method will work well for "small" data sets, but we're working harder than necessary, since we don't use the vectors U. This is a very accurate method, however.

Matlab using the Covariance

In this case, we will explicitly compute the covariance matrix, although some authors warn that this may introduce numerical error. We'll do it here just for comparison, and when p or n may be too large for Matlab to handle. For example, in dealing with photos or video, n may be in the hundreds of thousands, while p may only be in the hundreds. In that case, we may want to work with $X^T X$ (which would be $p \times p$) instead of X ($n \times p$). In that case, the Matlab code would look like the following:

```
[n,p]=size(X);
mx=mean(X,2);
Xm=X-repmat(mx,1,p);
C=(1/(p-1))*Xm'*Xm;
[V,S]=eig(C);
[V,S]=sorteig(V,S);
ss=diag(S)./sum(diag(S));
```

There is a problem here- the eigenvalues do not come back ordered. The `sorteig` command is not a built-in Matlab function- we have to write it ourselves. It's a function that will sort the eigenvalues and eigenvectors out for us:

```
function [V1,L1]=sorteig(V,L)
%Sorts the eigenvalues in L to descending order, re-orders
%the eigenvectors in V accordingly
lambda=diag(L);
n=length(lambda);
[val,idx]=sort(lambda,'descend');
L1=diag(val);
V1=V(:,idx);
```

Another way around this problem is not to use the `eig` command. We might, for example, use the `svd` command. In this case, we would have:

```
[p,n]=size(X);
mx=mean(X);
Xm=X-repmat(mx,p,1);
C=(1/p)*Xm'*Xm;
[V,S,V]=svd(C);
ss=diag(S)./sum(diag(S));
```

where the diagonal elements of S come out of the SVD in a decreasing fashion already. Note that the SVD of the covariance is the eigenvector decomposition...

In either case, we get the vectors U by using our relationship: $X\mathbf{v}_i = \sigma_i \mathbf{u}_i$:

```
U=X*V
```

The columns of U will not have unit length at this point, but they will be orthogonal. You would normalize them at this point, if desired.

0.7 A Second Proof (Optional)

We give a secondary argument that the first k principal components form the best $k-$dimensional subspace for the data (in terms of minimizing the MSE). We do not argue that this basis is the *only* basis for which this is true- indeed, as a $k-$dimensional subspace, any orthonormal basis will produce the same MSE.

First, we saw that, for any orthonormal basis $\{\boldsymbol{\psi}_i\}_{i=1}^n$

$$\frac{1}{p-1}\sum_{i=1}^{p}\|\boldsymbol{x}^{(i)}\| = \sum_{j=1}^{k}\boldsymbol{\psi}_j^T C\boldsymbol{\psi}_j + \sum_{j=k+1}^{n}\boldsymbol{\psi}_j^T C\boldsymbol{\psi}_j$$

and this sum is a constant. Therefore, minimizing the second term of the sum is the same as maximizing the first term of the sum. Using the eigenvector basis, we get that:

$$\sum_{j=1}^{k}\boldsymbol{\phi}_j^T C\boldsymbol{\phi}_j = \lambda_1 + \lambda_2 + \ldots + \lambda_k$$

and we can evaluate the second term directly as well:

$$\frac{1}{p-1}\sum_{i=1}^{p}\|\boldsymbol{x}_{\text{err}}^{(i)}\|^2 = \sum_{j=k+1}^{n}\boldsymbol{\phi}_j^T C\boldsymbol{\phi}_j = \lambda_{k+1} + \ldots + \lambda_n$$

We also stated that first k eigenvectors of C form the best basis over all $k-$term expansions, and we will prove that now:

Let $\{\boldsymbol{\psi}_i\}_{i=1}^k$ be any other $k-$dimensional basis. We can write each $\boldsymbol{\psi}_i$ in terms of its coordinates with respect to the eigenvectors of C,

$$\boldsymbol{\psi}_i = \sum_{k=1}^{n} \left(\boldsymbol{\psi}_i^T \boldsymbol{\phi}_k\right) \boldsymbol{\phi}_k = \Phi \boldsymbol{\alpha}_i$$

so that $\alpha_{ik} = \boldsymbol{\psi}_i^T \boldsymbol{\phi}_k$. Given this, we see that:

$$\boldsymbol{\psi}_i^T C \boldsymbol{\psi}_i = \boldsymbol{\alpha}_i^T \Lambda \boldsymbol{\alpha}_i$$

Now, consider the mean squared projection of the data onto this $k-$ dimensional basis:

$$\sum_{j=1}^{k} \boldsymbol{\psi}_j^T C \boldsymbol{\psi}_j = \sum_{j=1}^{k} \boldsymbol{\alpha}_j^T \Lambda \boldsymbol{\alpha}_j = \sum_{j=1}^{k} \lambda_1 \alpha_{j1}^2 + \ldots \lambda_n \alpha_{jn}^2$$

so that:

$$\sum_{j=1}^{k} \boldsymbol{\psi}_j^T C \boldsymbol{\psi}_j = \lambda_1 \sum_{j=1}^{k} \alpha_{j1}^2 + \lambda_2 \sum_{j=1}^{k} \alpha_{j2}^2 + \ldots \lambda_n \sum_{j=1}^{k} \alpha_{jn}^2$$

We're just about there- what we want to do now is to show that each of these coefficients is less than one.

The nifty trick is to consider the coefficients:

$$\{\alpha_{1i}, \ldots, \alpha_{ki}\} = \left\{\boldsymbol{\psi}_1^T \boldsymbol{\phi}_i, \ldots, \boldsymbol{\psi}_k^T \boldsymbol{\phi}_i\right\}$$

as the coefficients from the projection of $\boldsymbol{\phi}_i$ onto the subspace spanned by the $\boldsymbol{\psi}$'s:

$$\text{Proj}_\Psi(\boldsymbol{\phi}_i) = \alpha_{1i}\boldsymbol{\psi}_1 + \ldots + \alpha_{ki}\boldsymbol{\psi}_k$$

where we have what we need:

$$\sum_{j=1}^{k} \alpha_{ji}^2 = \|\text{Proj}_\Psi(\boldsymbol{\phi}_i)\|^2 \leq 1$$

with equality iff ϕ_i is in the span of the columns of Ψ. Now we can say the the maximum of:

$$\lambda_1 \sum_{j=1}^{k} \alpha_{j1}^2 + \lambda_2 \sum_{j=1}^{k} \alpha_{j2}^2 + \ldots \lambda_n \sum_{j=1}^{k} \alpha_{jn}^2$$

is found by replacing the coefficients of the first k terms to 1, and the remaining to zero. This corresponds, of course, to the value of the error found by using the eigenvector basis.

We have thus shown that the first k eigenvectors of the covariance matrix forms the best k dimensional subspace for the data- but notice that *any* basis for that k dimensional subspace would work.

0.8 Sunspot Analysis, Part I

In this example, we show one method for constructing a matrix from a time series, and from that matrix, we can extract the cyclical information from the data. This is intimately related to the *Fourier analysis* that we will discuss later- and in fact, under certain conditions, it can be shown that the best basis vectors of oscillatory data *are* the sinusoids that come out of the Fourier analysis. See [**?**] for more details on the method.

First, let us define some terms. We will say that a set of data, $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$ is translationally invariant if a shift of the data of the form:

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{bmatrix} \longrightarrow \begin{bmatrix} x_n \\ x_1 \\ x_2 \\ \vdots \\ x_{n-1} \end{bmatrix} \longrightarrow \begin{bmatrix} x_{n-1} \\ x_n \\ x_1 \\ \vdots \\ x_{n-2} \end{bmatrix} \longrightarrow \ldots$$

always forms another valid data point. A primary example is a signal time series of speech. Pieces of the data of length k from any initial time, t_0, all provide for equally valid samples of the voice.

In this example, we consider the sunspot data that comes with Matlab. This data represents a normalized number of sunspots, approximately every year from 1700 to 1987. The exact method of normalization is not important here- what is important is the relative cycling of the number of sunspots. Heavy sunspot activity (years with the peaks) causes problems with electronic communications on the earth, from cellular phones to cable TV.

There is a rule of thumb that says that the sunspot activity cycles approximately every 11 years. In this example, we will see how the KL eigenvectors correspond with this value- although remember that a full analysis will need to wait until we've discussed the Fourier analysis.

We first load the data and from this data we make a circulant matrix.

```
load sunspot.dat
npts=288;
t=sunspot(:,1);
x=sunspot(:,2);
plot(t,x);  %Do this if you haven't seen the data
meansun=mean(x);
x=x-meansun;  %Mean subtract the data

X=zeros(npts,npts);  %Will hold the matrix
X(:,1)=x;
idx=1:npts;

%Create the circulant matrix
for j=2:npts
   idx=[idx(npts) idx(1:npts-1)];
   X(:,j)=x(idx);
end

[U,S,V]=svd(X);
ss=diag(S)./sum(diag(S));
```

Type this into Matlab, and remark on the following:

- Plot the first 20 values of `ss`. Here you see that the singular values are doubled up, which corresponds to a doubling up of the eigenvalues of the covariance. Each of these two dimensional spaces correspond to a sinusoid of a certain period.
- To see the sinusoids, plot the first two eigenvectors, then the next two, then the next two. You'll see them line up like sine and cosine pairs for each period.
- The first two eigenvectors will correspond to the sine/cosine pair that best explain the data. It is this period that is approximately 11- Try to estimate the period from its graph (we will numerically estimate this period in the Fourier analysis section).

0.9 Eigenfaces

The application of PCA to face recognition seems to have begun in the late 1980's with the paper by Sirovich and Kirby [**?**, **?**]. Since then, there has been a quite extensive body of research in face recognition using similar techniques. A good starting point for locating additional resources and bibliographies are available on the internet; for example, the Face Recognition Homepage [**?**] and MIT's Artificial Intelligence Lab [**?**]. A recent text [**?**] also gives more details on PCA, its application to faces, and more applications to other data

analysis problems. We also note that current research in psychology and computer science is suggesting that perhaps the actual biological processes involved in familiarity are related to PCA [**?**].

In face recognition problems, we are given a large collection of photos of faces. In this example, we assume that all photos are the same size, that they are grayscale, that they have basically the same background, and have been manually centered (all eyes are at approximately the same location).

To be more specific, suppose we have 30 photos, and each has 262×294 pixels. Then each photo is a vector in $\mathbb{R}^{77,028}$. We look for a small dimensional basis, Φ, in which to represent the space of all faces in the database, so that if $\boldsymbol{x}^{(i)}$ is a face, it can be represented by its (much lower dimensional) coordinates, $\left[\boldsymbol{x}^{(i)}\right]_{\Phi}$.

We would like to use the PCA algorithm to compute this basis, but this is a very large eigenvector problem- The covariance matrix is 77028×77028. However, using the SVD we avoid computing the covariance matrix directly (which is good, since it can introduce rather large numerical errors).

Let X be the 77028×30 matrix formed from storing our photos, and suppose we have subtracted the mean "face" (as a vector in $\mathbb{R}^{77028}$). Then we find the best basis in $\mathbb{R}^{77028}$ via the SVD:

```
[U,S,V]=svd(X,'econ');
```

Be sure to include the `econ` option- That way, the matrix U will be 77028×30 instead of 77028×77028! The matrix S holds the singular values of X (a square, diagonal matrix), and V will be 30×30.

Our next step is to look at the dimensionality of the data- This means that we look at the graph of the (normalized) eigenvalues of the covariance to see if there are any natural gaps (as we discussed earlier). For visualization, we normally would include only 2 or 3 dimensions.

Either way, the columns of U are in $\mathbb{R}^{77,028}$, and we will choose only a "small" number of them. Here is the interesting translation: Since each eigenvector $\mathbf{u}_i$ is a vector in $\mathbb{R}^{77,028}$, *it can be realized as a photograph (face) since it has* 262×294 *pixels!* Thus the term "eigenface" was coined (I'm not sure who originally came up with it), and we get a set of orthonormal "faces" by which all faces in the database can be encoded. As an example, at least one commercial package uses 128 basis vectors [**?**].

Once the dimension has been set, we can numerically encode every face in terms of the weights of the linear combinations- this is the key to actual face recognition. You encode someone's face, and search the database for the closest match. Using the SVD, the "encoding" is the coordinate vector:

```
Coord=U(:,1:2)'*X;
```

This matrix multiplication has dimensions: $(2 \times 77028)(77028 \times 30)$, so the matrix `Coord` is 2×30 (for 30 points in $\mathbb{R}^2$). These are the "low dimensional representations" of the data, and so the faces can be plotted in $\mathbb{R}^2$.

Similarly, the reconstruction of the faces using only the two basis vectors can be computed as the following (do you see the projection?):

```
Recon=U(:,1:2)*Coord;
%To see the 4th reconstructed face:
image(reshape(Recon(:,4)+MeanFace),262,294)
```

The matrix `Recon` is 77028×30, and is the two dimensional approximation to the data in X.

0.10 A Movie Data Example

A movie is a collection of photographs, so that we can similarly construct a "best basis" for a movie segment. In this example, the author sat in front of an inexpensive webcam and tried to move in a periodic fashion. Figure 2 shows some sample frames from the movie[1] A script was then written to load each frame into Matlab. In this case, we have 109 frames, and each frame has $120 \times 160 = 19200$ pixels, and the movie is in grayscale.

[1] As a technical note, the webcam recorded the motion in AVI format, which was subsequently converted to MPEG and then split using software that is freely available on the internet.

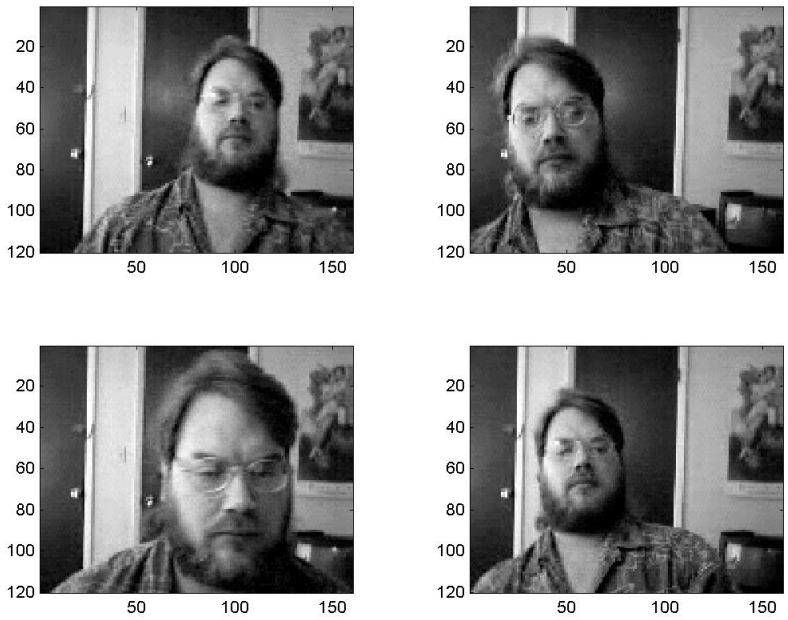

Figure 2: Four sample frames from the 109 frame movie. Each frame is 120×160.

To reconstruct the movie in Matlab, use the following commands:

```
load author1 %This is a matrix Y1 that is 19,200 x 109

for j=1:109
   A=reshape(Y1(:,j),120,160);
   imagesc(A);
   if j==1
      colormap(gray);
   end
   M(j)=getframe;
end
```

To replay the movie, type `movie(M)`.

Exercise: Find the best basis for the movie clip. Plot the mean image and the first five eigenvectors (as movie frames or images). Plot the movie in the best three dimensional coordinate system. Comment on what you see- for example, is the movie actually periodic? What kinds of features are being encoded in the eigenvectors?