A TUTORIAL ON PRINCIPAL COMPONENT ANALYSIS
Derivation, Discussion and Singular Value Decomposition

Jon Shlens | *jonshlens@ucsd.edu*

25 March 2003 | Version 1

Principal component analysis (PCA) is a mainstay of modern data analysis - a black box that is widely used but poorly understood. The goal of this paper is to dispel the magic behind this black box. This tutorial focuses on building a solid intuition for how and why principal component analysis works; furthermore, it crystallizes this knowledge by deriving from first principals, the mathematics behind PCA . This tutorial does not shy away from explaining the ideas informally, nor does it shy away from the mathematics. The hope is that by addressing both aspects, readers of all levels will be able to gain a better understanding of the power of PCA as well as the when, the how and the why of applying this technique.

1 Overview

Principal component analysis (*PCA*) has been called one of the most valuable results from applied linear algebra. *PCA* is used abundantly in all forms of analysis - from neuroscience to computer graphics - because it is a simple, non-parametric method of extracting relevant information from confusing data sets. With minimal additional effort *PCA* provides a roadmap for how to reduce a complex data set to a lower dimension to reveal the sometimes hidden, simplified dynamics that often underlie it.

The goal of this tutorial is to provide both an intuitive feel for *PCA*, and a thorough discussion of this topic. We will begin with a simple example and provide an intuitive explanation of the goal of *PCA*. We will continue by adding mathematical rigor to place it within the framework of linear algebra and explicitly solve this problem. We will see how and why *PCA* is intimately related to the mathematical technique of singular value decomposition (*SVD*). This understanding will lead us to a prescription for how to apply *PCA* in the real world. We will discuss both the assumptions behind this technique as well as possible extensions to overcome these limitations.

The discussion and explanations in this paper are informal in the spirit of a tutorial. The goal of this paper is to *educate*. Occasionally, rigorous mathematical proofs are necessary although relegated to the Appendix. Although not as vital to the tutorial, the proofs are presented for the adventurous reader who desires a more complete understanding of the math. The only assumption is that the reader has a working knowledge of linear algebra. Nothing more. Please feel free to contact me with any suggestions, corrections or comments.

2 Motivation: A Toy Example

Here is the perspective: we are an experimenter. We are trying to understand some phenomenon by measuring various quantities (e.g. spectra, voltages, velocities, etc.) in our system. Unfortunately, we can not figure out what is happening because the data appears clouded, unclear and even redundant. This is not a trivial problem, but rather a fundamental obstacle to experimental science. Examples abound from complex systems such as neuroscience, photoscience, meteorology and oceanography - the number of variables to measure can be unwieldy and at times even *deceptive*, because the underlying dynamics can often be quite simple.

Take for example a simple toy problem from physics diagrammed in Figure 1. Pretend we are studying the motion of the physicist's ideal spring. This system consists of a ball of mass m attached to a *massless, frictionless* spring. The ball is released a small distance away from equilibrium (i.e. the spring is stretched). Because the spring is "ideal," it oscillates indefinitely along the x-axis about its equilibrium at a set frequency.

This is a standard problem in physics in which the

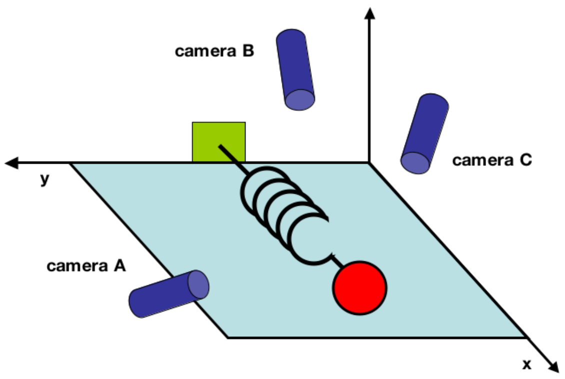


Figure 1: A diagram of the toy example.

motion along the x direction is solved by an explicit function of time. In other words, the underlying dynamics can be expressed as a function of a single variable x.

However, being ignorant experimenters we do not know any of this. We do not know which, let alone how many, axes and dimensions are important to measure. Thus, we decide to measure the ball's position in a three-dimensional space (since we live in a three dimensional world). Specifically, we place three movie cameras around our system of interest. At 200 Hz each movie camera records an image indicating a two dimensional position of the ball (a projection). Unfortunately, because of our ignorance, we do not even know what are the *real* "x", "y" and "z" axes, so we choose three camera axes $\{\vec{\mathbf{a}}, \vec{\mathbf{b}}, \vec{\mathbf{c}}\}$ at some arbitrary angles with respect to the system. The angles between our measurements might not even be 90^o! Now, we record with the cameras for 2 minutes. The big question remains: *how do we get from this data set to a simple equation of x?*

We know a-priori that if we were smart experimenters, we would have just measured the position along the x-axis with one camera. But this is not what happens in the real world. We often do not know what measurements best reflect the dynamics of our system in question. Furthermore, we sometimes record more dimensions than we actually need!

Also, we have to deal with that pesky, real-world problem of *noise*. In the toy example this means that we need to deal with air, imperfect cameras or even friction in a less-than-ideal spring. Noise contaminates our data set only serving to obfuscate the dynamics further. *This toy example is the challenge experimenters face everyday.* We will refer to this example as we delve further into abstract concepts. Hopefully, by the end of this paper we will have a good understanding of how to systematically extract x using principal component analysis.

3 Framework: Change of Basis

The Goal: Principal component analysis computes the most meaningful *basis* to re-express a noisy, garbled data set. The hope is that this new basis will filter out the noise and reveal hidden dynamics. In the example of the spring, the explicit goal of *PCA* is to determine: "the dynamics are along the x-axis." In other words, the goal of *PCA* is to determine that $\hat{\mathbf{x}}$ - the unit basis vector along the x-axis - is the important dimension. Determining this fact allows an experimenter to discern which dynamics are important and which are just redundant.

3.1 A Naive Basis

With a more precise definition of our goal, we need a more precise definition of our data as well. For each time sample (or experimental trial), an experimenter records a set of data consisting of multiple measurements (e.g. voltage, position, etc.). The number of measurement types is the *dimension* of the data set. In the case of the spring, this data set has 12,000 6-dimensional vectors, where each camera contributes a 2-dimensional projection of the ball's position.

In general, each data sample is a vector in m-dimensional space, where m is the number of measurement types. Equivalently, every time sample is a vector that lies in an m-dimensional *vector space* spanned by an orthonormal basis. All measurement vectors in this space are a linear combination of this set of unit length basis vectors. A naive and simple choice of a basis $\mathbf{B}$ is the identity matrix $\mathbf{I}$.

$$\mathbf{B} = \begin{bmatrix} \mathbf{b_1} \\ \mathbf{b_2} \\ \vdots \\ \mathbf{b_m} \end{bmatrix} = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \end{bmatrix} = \mathbf{I}$$

where each *row* is a basis vector $\mathbf{b}_i$ with m components.

To summarize, at one point in time, camera A records a corresponding position $(x_A(t), y_A(t))$. Each trial can be expressed as a six dimesional column vector $\vec{X}$.

$$\vec{X} = \begin{bmatrix} x_A \\ y_A \\ x_B \\ y_B \\ x_C \\ y_C \end{bmatrix}$$

where each camera contributes two points and the entire vector $\vec{X}$ is the set of coefficients in the naive basis $\mathbf{B}$.

3.2 Change of Basis

With this rigor we may now state more precisely what *PCA* asks: *Is there another basis, which is a linear combination of the original basis, that best re-expresses our data set?*

A close reader might have noticed the conspicuous addition of the word *linear*. Indeed, *PCA* makes one stringent but powerful assumption: *linearity*. Linearity vastly simplifies the problem by (1) restricting the set of potential bases, and (2) formalizing the implicit assumption of continuity in a data set. *A subtle point it is, but we have already assumed linearity by implicitly stating that the data set even characterizes the dynamics of the system!* In other words, we are already relying on the superposition principal of linearity to believe that the data characterizes or provides an ability to interpolate between the individual data points[1].

[1]To be sure, complex systems are almost always nonlinear and often their main qualitative features are a direct result of their nonlinearity. However, locally linear approximations usually provide a good approximation because non-linear, higher order terms vanish at the limit of small perturbations.

With this assumption *PCA* is now limited to re-expressing the data as a *linear combination* of its basis vectors. Let $\mathbf{X}$ and $\mathbf{Y}$ be $m \times n$ matrices related by a linear transformation $\mathbf{P}$. $\mathbf{X}$ is the original recorded data set and $\mathbf{Y}$ is a re-representation of that data set.

$$\mathbf{PX} = \mathbf{Y} \tag{1}$$

Also let us define the following quantities[2].

- $\mathbf{p_i}$ are the *rows* of $\mathbf{P}$
- $\mathbf{x_i}$ are the *columns* of $\mathbf{X}$
- $\mathbf{y_i}$ are the *columns* of $\mathbf{Y}$.

Equation 1 represents a *change of basis* and thus can have many interpretations.

1. $\mathbf{P}$ is a matrix that transforms $\mathbf{X}$ into $\mathbf{Y}$.
2. Geometrically, $\mathbf{P}$ is a rotation and a stretch which again transforms $\mathbf{X}$ into $\mathbf{Y}$.
3. The rows of $\mathbf{P}$, $\{\mathbf{p_1}, \ldots, \mathbf{p_m}\}$, are a set of new basis vectors for expressing the *columns* of $\mathbf{X}$.

The latter interpretation is not obvious but can be seen by writing out the explicit dot products of $\mathbf{PX}$.

$$\begin{aligned} \mathbf{PX} &= \begin{bmatrix} \mathbf{p_1} \\ \vdots \\ \mathbf{p_m} \end{bmatrix} \begin{bmatrix} \mathbf{x_1} & \cdots & \mathbf{x_n} \end{bmatrix} \\ \mathbf{Y} &= \begin{bmatrix} \mathbf{p_1 \cdot x_1} & \cdots & \mathbf{p_1 \cdot x_n} \\ \vdots & \ddots & \vdots \\ \mathbf{p_m \cdot x_1} & \cdots & \mathbf{p_m \cdot x_n} \end{bmatrix} \end{aligned}$$

We can note the form of each column of $\mathbf{Y}$.

$$\mathbf{y}_i = \begin{bmatrix} \mathbf{p_1 \cdot x_i} \\ \vdots \\ \mathbf{p_m \cdot x_i} \end{bmatrix}$$

We recognize that each coefficient of $\mathbf{y_i}$ is a dot-product of $\mathbf{x_i}$ with the corresponding row in $\mathbf{P}$. In

[2]In this section $\mathbf{x_i}$ and $\mathbf{y_i}$ are *column* vectors, but be forewarned. In all other sections $\mathbf{x_i}$ and $\mathbf{y_i}$ are *row* vectors.

other words, the j^{th} coefficient of $\mathbf{y_i}$ is a projection on to the j^{th} row of $\mathbf{P}$. This is in fact the very form of an equation where $\mathbf{y_i}$ is a projection on to the basis of $\{\mathbf{p_1}, \ldots, \mathbf{p_m}\}$. Therefore, the *rows* of $\mathbf{P}$ are indeed a new set of basis vectors for representing of *columns* of $\mathbf{X}$.

3.3 Questions Remaining

By assuming linearity the problem reduces to finding the appropriate *change of basis.* The row vectors $\{\mathbf{p_1}, \ldots, \mathbf{p_m}\}$ in this transformation will become the *principal components* of $\mathbf{X}$. Several questions now arise.

- What is the best way to "re-express" $\mathbf{X}$?
- What is a good choice of basis $\mathbf{P}$?

These questions must be answered by next asking ourselves what features we would like $\mathbf{Y}$ *to exhibit.* Evidently, additional assumptions beyond *linearity* are required to arrive at a reasonable result. The selection of these assumptions is the subject of the next section.

4 Variance and the Goal

Now comes the most important question: *what does "best express" the data mean?* This section will build up an intuitive answer to this question and along the way tack on additional assumptions. The end of this section will conclude with a mathematical goal for deciphering "garbled" data.

In a linear system "garbled" can refer to only two potential confounds: *noise* and *redundancy.* Let us deal with each situation individually.

4.1 Noise

Noise in any data set must be low or - no matter the analysis technique - no information about a system can be extracted. There exists no absolute scale for noise but rather all noise is measured relative to the measurement. A common measure is the the *signal-to-noise ratio* (*SNR*), or a ratio of variances σ^2.

$$SNR = \frac{\sigma^2_{signal}}{\sigma^2_{noise}} \tag{2}$$

A high *SNR* ($\gg 1$) indicates high precision data, while a low *SNR* indicates noise contaminated data.

Pretend we plotted all data from camera A from the spring in Figure 2. Any individual camera should record motion in a straight line. Therefore, any spread deviating from straight-line motion must be noise. The variance due to the signal and noise are indicated in the diagram graphically. The ratio of the two, the *SNR*, thus measures how "fat" the oval is - a range of possiblities include a thin line (*SNR* $\gg 1$), a perfect circle (*SNR* $= 1$) or even worse. In summary, we must assume that our measurement devices are reasonably good. Quantitatively, this corresponds to a high *SNR*.

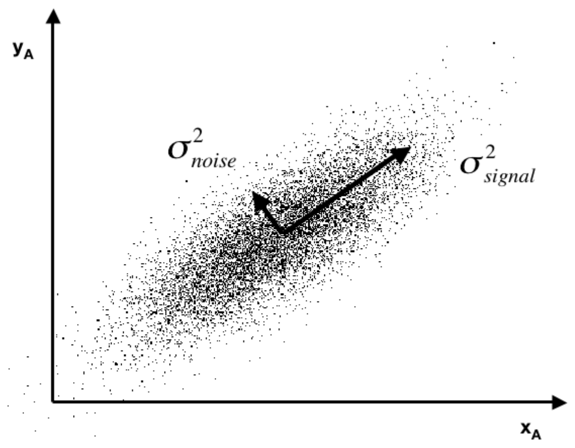


Figure 2: A simulated plot of (x_A, y_A) for camera A. The signal and noise variances σ^2_{signal} and σ^2_{noise} are graphically represented.

4.2 Redundancy

Redundancy is a more tricky issue. This issue is particularly evident in the example of the spring. In this case multiple sensors record the same dynamic

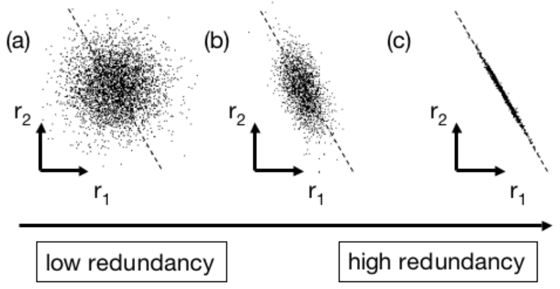


Figure 3: A spectrum of possible redundancies in data from the two separate recordings r_1 and r_2 (e.g. x_A, y_B). The best-fit line $r_2 = kr_1$ is indicated by the dashed line.

information. Consider Figure 3 as a range of possibile plots between two arbitrary measurement types r_1 and r_2. Panel (a) depicts two recordings with no redundancy. In other words, r_1 is entirely uncorrelated with r_2. This situation could occur by plotting two variables such as $(x_A, humidity)$.

However, in panel (c) both recordings appear to be strongly related. This extremity might be achieved by several means.

- A plot of $(x_A, \tilde{x}_A)$ where x_A is in meters and $\tilde{x}_A$ is in inches.
- A plot of (x_A, x_B) if cameras A and B are very nearby.

Clearly in panel (c) it would be more meaningful to just have recorded a single variable, the linear combination $r_2 - kr_1$, instead of two variables r_1 and r_2 separately. Recording solely the linear combination $r_2 - kr_1$ would both express the data more concisely and reduce the number of sensor recordings ($2 \rightarrow 1$ variables). Indeed, this is the very idea behind dimensional reduction.

4.3 Covariance Matrix

The *SNR* is solely determined by calculating variances. A corresponding but simple way to quantify the redundancy between individual recordings is to calculate something like the variance. I say "something like" because the variance is the spread due to one variable but we are concerned with the spread between variables.

Consider two sets of simultaneous measurements with zero mean[3].

$$A = \{a_1, a_2, \ldots, a_n\} \ , \ B = \{b_1, b_2, \ldots, b_n\}$$

The variance of A and B are individually defined as follows.

$$\sigma_A^2 = \langle a_i a_i \rangle_i \ , \ \sigma_B^2 = \langle b_i b_i \rangle_i$$

where the expectation[4] is the average over n variables. The *covariance* between A and B is a straighforward generalization.

$$covariance\ of\ A\ and\ B \equiv \sigma_{AB}^2 = \langle a_i b_i \rangle_i$$

Two important facts about the covariance.

- $\sigma_{AB}^2 = 0$ if and only if A and B are entirely uncorrelated.
- $\sigma_{AB}^2 = \sigma_A^2$ if $A = B$.

We can equivalently convert the sets of A and B into corresponding row vectors.

$$\begin{aligned} \mathbf{a} &= [a_1\ a_2\ \ldots\ a_n] \\ \mathbf{b} &= [b_1\ b_2\ \ldots\ b_n] \end{aligned}$$

so that we may now express the covariance as a dot product matrix computation.

$$\sigma_{\mathbf{ab}}^2 \equiv \frac{1}{n-1}\mathbf{a}\mathbf{b}^T \qquad (3)$$

where the beginning term is a constant for normalization[5].

Finally, we can generalize from two vectors to an arbitrary number. We can rename the row vectors $\mathbf{x_1} \equiv \mathbf{a}$, $\mathbf{x_2} \equiv \mathbf{b}$ and consider additional indexed row

[3]These data sets are in *mean deviation form* because the means have been subtracted off or are zero.

[4]$\langle \cdot \rangle_i$ denotes the average over values indexed by i.

[5]The simplest possible normalization is $\frac{1}{n}$. However, this provides a biased estimation of variance particularly for small n. It is beyond the scope of this paper to show that the proper normalization for an unbiased estimator is $\frac{1}{n-1}$.

vectors $\mathbf{x_3}, \ldots, \mathbf{x_m}$. Now we can define a new $m \times n$ matrix $\mathbf{X}$.

$$\mathbf{X} = \begin{bmatrix} \mathbf{x_1} \\ \vdots \\ \mathbf{x_m} \end{bmatrix} \tag{4}$$

One interpretation of $\mathbf{X}$ is the following. Each *row* of $\mathbf{X}$ corresponds to all measurements of a particular type ($\mathbf{x_i}$). Each *column* of $\mathbf{X}$ corresponds to a set of measurements from one particular trial (this is $\vec{X}$ from section 3.1). We now arrive at a definition for the *covariance matrix* $\mathbf{S_X}$.

$$\mathbf{S_X} \equiv \frac{1}{n-1}\mathbf{X}\mathbf{X}^T \tag{5}$$

Consider how the matrix form $\mathbf{X}\mathbf{X}^T$, in that order, computes the desired value for the ij^{th} element of $\mathbf{S_X}$. Specifically, the ij^{th} element of the variance is the dot product between the vector of the i^{th} measurement type with the vector of the j^{th} measurement type.

- The ij^{th} value of $\mathbf{X}\mathbf{X}^T$ is equivalent to substituting $\mathbf{x_i}$ and $\mathbf{x_j}$ into Equation 3.

- $\mathbf{S_X}$ is a square symmetric $m \times m$ matrix.

- The diagonal terms of $\mathbf{S_X}$ are the *variance* of particular measurement types.

- The off-diagonal terms of $\mathbf{S_X}$ are the *covariance* between measurement types.

Computing $\mathbf{S_X}$ quantifies the correlations between all possible pairs of measurements. Between one pair of measurements, a large covariance corresponds to a situation like panel (c) in Figure 3, while zero covariance corresponds to entirely uncorrelated data as in panel (a).

$\mathbf{S_X}$ is special. The covariance matrix describes all relationships between pairs of measurements in our data set. Pretend we have the option of manipulating $\mathbf{S_X}$. We will suggestively define our manipulated covariance matrix $\mathbf{S_Y}$. What features do we want to optimize in $\mathbf{S_Y}$?

4.4 Diagonalize the Covariance Matrix

If our goal is to reduce redundancy, then we would like each variable to co-vary as little as possible with other variables. More precisely, to remove redundancy we would like the covariances between separate measurements to be zero. What would the optimized covariance matrix $\mathbf{S_Y}$ look like? Evidently, in an "optimized" matrix all off-diagonal terms in $\mathbf{S_Y}$ are zero. Therefore, removing redundancy diagnolizes $\mathbf{S_Y}$.

There are many methods for diagonalizing $\mathbf{S_Y}$. It is curious to note that *PCA* arguably selects the easiest method, perhaps accounting for its widespread application.

PCA assumes that all basis vectors $\{\mathbf{p_1}, \ldots, \mathbf{p_m}\}$ are orthonormal (i.e. $\mathbf{p_i} \cdot \mathbf{p_j} = \delta_{ij}$). In the language of linear algebra, *PCA* assumes $\mathbf{P}$ is an orthonormal matrix. Secondly *PCA* assumes the directions with the largest variances are the most "important" or in other words, most *principal.* Why are these assumptions easiest?

Envision how *PCA* might work. By the variance assumption *PCA* first selects a normalized direction in m-dimensional space along which the variance in $\mathbf{X}$ is maximized - it saves this as $\mathbf{p_1}$. Again it finds another direction along which variance is maximized, however, because of the orthonormality condition, it restricts its search to all directions perpindicular to all previous selected directions. This could continue until m directions are selected. The resulting ordered set of $\mathbf{p}$'s are the *principal components.*

In principle this simple pseudo-algorithm works, however that would bely the true reason why the orthonormality assumption is particularly judicious. Namely, the true benefit to this assumption is that *it makes the solution amenable to linear algebra.* There exist decompositions that can provide efficient, explicit algebraic solutions.

Notice what we also gained with the second assumption. We have a method for judging the "importance" of the each prinicipal direction. Namely, the variances associated with each direction $\mathbf{p_i}$ quantify how principal each direction is. We could thus rank-order each basis vector $\mathbf{p_i}$ according to the corresponding variances.

We will now pause to review the implications of all the assumptions made to arrive at this mathematical goal.

4.5 Summary of Assumptions and Limits

This section provides an important context for understanding when *PCA* might perform poorly as well as a road map for understanding new extensions to *PCA*. The final section provides a more lengthy discussion about recent extensions.

I. *Linearity*
Linearity frames the problem as a change of basis. Several areas of research have explored how applying a nonlinearity prior to performing *PCA* could extend this algorithm - this has been termed *kernel PCA*.

II. *Mean and variance are sufficient statistics.*
The formalism of *sufficient statistics* captures the notion that the mean and the variance entirely describe a probability distribution. The only zero-mean probability distribution that is fully described by the variance is the Gaussian distribution.

In order for this assumption to hold, the probability distribution of $\mathbf{x_i}$ must be Gaussian. Deviations from a Gaussian could invalidate this assumption[6]. On the flip side, this assumption formally guarantees that the *SNR* and the covariance matrix fully characterize the noise and redundancies.

III. *Large variances have important dynamics.*
This assumption also encompasses the belief that the data has a high *SNR*. Hence, principal components with larger associated variances represent interesting dynamics, while those with lower variancees represent noise.

IV. *The principal components are orthogonal.*
This assumption provides an intuitive simplification that makes *PCA* soluble with linear algebra decomposition techniques. These techniques are highlighted in the two following sections.

We have discussed all aspects of deriving *PCA* - what remains is the linear algebra solutions. The first solution is somewhat straightforward while the second solution involves understanding an important decomposition.

5 Solving PCA: Eigenvectors of Covariance

We will derive our first algebraic solution to *PCA* using linear algebra. This solution is based on an important property of eigenvector decomposition. Once again, the data set is $\mathbf{X}$, an $m \times n$ matrix, where m is the number of measurement types and n is the number of data trials. The goal is summarized as follows.

> Find some orthonormal matrix $\mathbf{P}$ where $\mathbf{Y} = \mathbf{PX}$ such that $\mathbf{S_Y} \equiv \frac{1}{n-1}\mathbf{YY}^T$ is diagonalized. The rows of $\mathbf{P}$ are the *principal components* of $\mathbf{X}$.

We begin by rewriting $\mathbf{S_Y}$ in terms of our variable of choice $\mathbf{P}$.

$$\begin{aligned}
\mathbf{S_Y} &= \frac{1}{n-1}\mathbf{YY}^T \\
&= \frac{1}{n-1}(\mathbf{PX})(\mathbf{PX})^T \\
&= \frac{1}{n-1}\mathbf{PXX}^T\mathbf{P}^T \\
&= \frac{1}{n-1}\mathbf{P}(\mathbf{XX}^T)\mathbf{P}^T \\
\mathbf{S_Y} &= \frac{1}{n-1}\mathbf{PAP}^T
\end{aligned}$$

[6]**A sidebar question:** What if $\mathbf{x}_i$ are not Gaussian distributed? Diagonalizing a covariance matrix might not produce satisfactory results. The most rigorous form of removing redundancy is statistical independence.

$$P(\mathbf{y}_1, \mathbf{y}_2) = P(\mathbf{y}_1)P(\mathbf{y}_2)$$

where $P(\cdot)$ denotes the probability density. The class of algorithms that attempt to find the $\mathbf{y_1}, \ , \ldots, \ \mathbf{y}_m$ that satisfy this statistical constraint are termed independent component analysis (*ICA*). In practice though, quite a lot of real world data are Gaussian distributed - thanks to the Central Limit Theorem - and *PCA* is usually a robust solution to slight deviations from this assumption.

Note that we defined a new matrix $\mathbf{A} \equiv \mathbf{X}\mathbf{X}^T$, where $\mathbf{A}$ is *symmetric* (by Theorem 2 of Appendix A).

Our roadmap is to recognize that a symmetric matrix ($\mathbf{A}$) is diagonalized by an orthogonal matrix of its eigenvectors (by Theorems 3 and 4 from Appendix A). For a symmetric matrix $\mathbf{A}$ Theorem 4 provides:

$$\mathbf{A} = \mathbf{E}\mathbf{D}\mathbf{E}^T \tag{6}$$

where $\mathbf{D}$ is a diagonal matrix and $\mathbf{E}$ is a matrix of eigenvectors of $\mathbf{A}$ arranged as columns.

The matrix $\mathbf{A}$ has $r \leq m$ orthonormal eigenvectors where r is the rank of the matrix. The rank of $\mathbf{A}$ is less than m when $\mathbf{A}$ is *degenerate* or all data occupy a subspace of dimension $r \leq m$. Maintaining the constraint of orthogonality, we can remedy this situation by selecting $(m - r)$ additional orthonormal vectors to "fill up" the matrix $\mathbf{E}$. These additional vectors do not effect the final solution because the variances associated with these directions are zero.

Now comes the trick. *We select the matrix* $\mathbf{P}$ *to be a matrix where each row* $\mathbf{p_i}$ *is an eigenvector of* $\mathbf{X}\mathbf{X}^T$. By this selection, $\mathbf{P} \equiv \mathbf{E^T}$. Substituting into Equation 6, we find $\mathbf{A} = \mathbf{P}^T\mathbf{D}\mathbf{P}$. With this relation and Theorem 1 of Appendix A ($\mathbf{P}^{-1} = \mathbf{P}^T$) we can finish evaluating $\mathbf{S_Y}$.

$$\begin{aligned}
\mathbf{S_Y} &= \frac{1}{n-1}\mathbf{P}\mathbf{A}\mathbf{P}^T \\
&= \frac{1}{n-1}\mathbf{P}(\mathbf{P}^T\mathbf{D}\mathbf{P})\mathbf{P}^T \\
&= \frac{1}{n-1}(\mathbf{P}\mathbf{P}^T)\mathbf{D}(\mathbf{P}\mathbf{P}^T) \\
&= \frac{1}{n-1}(\mathbf{P}\mathbf{P}^{-1})\mathbf{D}(\mathbf{P}\mathbf{P}^{-1}) \\
\mathbf{S_Y} &= \frac{1}{n-1}\mathbf{D}
\end{aligned}$$

It is evident that the choice of $\mathbf{P}$ diagonalizes $\mathbf{S_Y}$. This was the goal for *PCA*. We can summarize the results of *PCA* in the matrices $\mathbf{P}$ and $\mathbf{S_Y}$.

- The principal components of $\mathbf{X}$ are the eigenvectors of $\mathbf{X}\mathbf{X}^T$; or the rows of $\mathbf{P}$.
- The i^{th} diagonal value of $\mathbf{S_Y}$ is the variance of $\mathbf{X}$ along $\mathbf{p_i}$.

In practice computing *PCA* of a data set $\mathbf{X}$ entails (1) subtracting off the mean of each measurement type and (2) computing the eigenvectors of $\mathbf{X}\mathbf{X}^T$. This solution is encapsulated in demonstration Matlab code included in Appendix B.

6 A More General Solution: Singular Value Decomposition

This section is the most mathematically involved and can be skipped without much loss of continuity. It is presented solely for completeness. We derive another algebraic solution for *PCA* and in the process, find that *PCA* is closely related to singular value decomposition (*SVD*). In fact, the two are so intimately related that the names are often used interchangeably. What we will see though is that *SVD* is a more general method of understanding *change of basis.*

We begin by quickly deriving the decomposition. In the following section we interpret the decomposition and in the last section we relate these results to *PCA*.

6.1 Singular Value Decomposition

Let $\mathbf{X}$ be an arbitrary $n \times m$ matrix[7] and $\mathbf{X}^T\mathbf{X}$ be a rank r, square, symmetric $n \times n$ matrix. In a seemingly unmotivated fashion, let us define all of the quantities of interest.

- $\{\hat{\mathbf{v}}_1, \hat{\mathbf{v}}_2, \ldots, \hat{\mathbf{v}}_r\}$ is the set of *orthonormal* $m \times 1$ eigenvectors with associated eigenvalues $\{\lambda_1, \lambda_2, \ldots, \lambda_r\}$ for the symmetric matrix $\mathbf{X}^T\mathbf{X}$.

$$(\mathbf{X}^T\mathbf{X})\hat{\mathbf{v}}_i = \lambda_i\hat{\mathbf{v}}_i$$

- $\sigma_i \equiv \sqrt{\lambda_i}$ are positive real and termed the *singular values.*
- $\{\hat{\mathbf{u}}_1, \hat{\mathbf{u}}_2, \ldots, \hat{\mathbf{u}}_r\}$ is the set of *orthonormal* $n \times 1$ vectors defined by $\hat{\mathbf{u}}_\mathbf{i} \equiv \frac{\mathbf{1}}{\sigma_\mathbf{i}}\mathbf{X}\hat{\mathbf{v}}_\mathbf{i}$.

We obtain the final definition by referring to Theorem 5 of Appendix A. The final definition includes two new and unexpected properties.

[7]Notice that in this section only we are reversing convention from $m \times n$ to $n \times m$. The reason for this derivation will become clear in section 6.3.

- $\hat{\mathbf{u}}_\mathbf{i} \cdot \hat{\mathbf{u}}_\mathbf{j} = \delta_{ij}$
- $\|\mathbf{X}\hat{\mathbf{v}}_\mathbf{i}\| = \sigma_i$

These properties are both proven in Theorem 5. We now have all of the pieces to construct the decomposition. The "value" version of singular value decomposition is just a restatement of the third definition.

$$\mathbf{X}\hat{\mathbf{v}}_i = \sigma_i \hat{\mathbf{u}}_i \tag{7}$$

This result says a quite a bit. $\mathbf{X}$ multiplied by an eigenvector of $\mathbf{X}^T\mathbf{X}$ is equal to a scalar times another vector. The set of eigenvectors $\{\hat{\mathbf{v}}_1, \hat{\mathbf{v}}_2, \ldots, \hat{\mathbf{v}}_r\}$ and the set of vectors $\{\hat{\mathbf{u}}_1, \hat{\mathbf{u}}_2, \ldots, \hat{\mathbf{u}}_r\}$ are both orthonormal sets or bases in r-dimensional space.

We can summarize this result for all vectors in one matrix multiplication by following the prescribed construction in Figure 4. We start by constructing a new diagonal matrix $\mathbf{\Sigma}$.

$$\mathbf{\Sigma} \equiv \begin{bmatrix} \sigma_{\tilde{1}} & & & & & \\ & \ddots & & & \mathbf{0} & \\ & & \sigma_{\tilde{r}} & & & \\ & & & 0 & & \\ & \mathbf{0} & & & \ddots & \\ & & & & & 0 \end{bmatrix}$$

where $\sigma_{\tilde{1}} \geq \sigma_{\tilde{2}} \geq \ldots \geq \sigma_{\tilde{r}}$ are the rank-ordered set of singular values. Likewise we construct accompanying orthogonal matrices $\mathbf{V}$ and $\mathbf{U}$.

$$\begin{aligned} \mathbf{V} &= [\hat{\mathbf{v}}_{\tilde{\mathbf{1}}}\ \hat{\mathbf{v}}_{\tilde{\mathbf{2}}}\ \ldots\ \hat{\mathbf{v}}_{\tilde{\mathbf{m}}}] \\ \mathbf{U} &= [\hat{\mathbf{u}}_{\tilde{\mathbf{1}}}\ \hat{\mathbf{u}}_{\tilde{\mathbf{2}}}\ \ldots\ \hat{\mathbf{u}}_{\tilde{\mathbf{n}}}] \end{aligned}$$

where we have appended an additional $(m-r)$ and $(n-r)$ orthonormal vectors to "fill up" the matrices for $\mathbf{V}$ and $\mathbf{U}$ respectively[8]. Figure 4 provides a graphical representation of how all of the pieces fit together to form the matrix version of *SVD*.

$$\mathbf{XV} = \mathbf{U\Sigma} \tag{8}$$

where each column of $\mathbf{V}$ and $\mathbf{U}$ perform the "value" version of the decomposition (Equation 7). Because $\mathbf{V}$ is orthogonal, we can multiply both sides by $\mathbf{V}^{-1} = \mathbf{V}^T$ to arrive at the final form of the decomposition.

$$\mathbf{X} = \mathbf{U\Sigma V}^T \tag{9}$$

Although it was derived without motivation, this decomposition is quite powerful. Equation 9 states that *any* arbitrary matrix $\mathbf{X}$ can be converted to an orthogonal matrix, a diagonal matrix and another orthogonal matrix (or a rotation, a stretch and a second rotation). Making sense of Equation 9 is the subject of the next section.

[8]This is the same procedure used to fixe the degeneracy in the previous section.

6.2 Interpreting SVD

The final form of *SVD* (Equation 9) is a concise but thick statement to understand. Let us instead reinterpret Equation 7.

$$\mathbf{Xa} = k\mathbf{b}$$

where $\mathbf{a}$ and $\mathbf{b}$ are column vectors and k is a scalar constant. The set $\{\hat{\mathbf{v}}_\mathbf{1}, \hat{\mathbf{v}}_\mathbf{2}, \ldots, \hat{\mathbf{v}}_\mathbf{m}\}$ is analogous to $\mathbf{a}$ and the set $\{\hat{\mathbf{u}}_\mathbf{1}, \hat{\mathbf{u}}_\mathbf{2}, \ldots, \hat{\mathbf{u}}_\mathbf{n}\}$ is analogous to $\mathbf{b}$. What is unique though is that $\{\hat{\mathbf{v}}_\mathbf{1}, \hat{\mathbf{v}}_\mathbf{2}, \ldots, \hat{\mathbf{v}}_\mathbf{m}\}$ and $\{\hat{\mathbf{u}}_\mathbf{1}, \hat{\mathbf{u}}_\mathbf{2}, \ldots, \hat{\mathbf{u}}_\mathbf{n}\}$ are orthonormal sets of vectors which *span* an m or n dimensional space, respectively. In particular, loosely speaking these sets appear to span all possible "inputs" ($\mathbf{a}$) and "outputs" ($\mathbf{b}$). Can we formalize the view that $\{\hat{\mathbf{v}}_\mathbf{1}, \hat{\mathbf{v}}_\mathbf{2}, \ldots, \hat{\mathbf{v}}_\mathbf{n}\}$ and $\{\hat{\mathbf{u}}_\mathbf{1}, \hat{\mathbf{u}}_\mathbf{2}, \ldots, \hat{\mathbf{u}}_\mathbf{n}\}$ span all possible "inputs" and "outputs"?

We can manipulate Equation 9 to make this fuzzy hypothesis more precise.

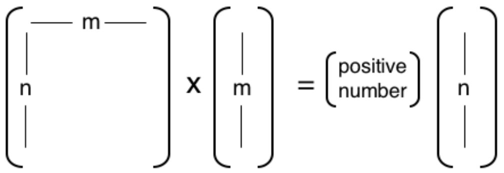
$$\begin{aligned} \mathbf{X} &= \mathbf{U\Sigma V}^T \\ \mathbf{U}^T\mathbf{X} &= \mathbf{\Sigma V}^T \\ \mathbf{U}^T\mathbf{X} &= \mathbf{Z} \end{aligned}$$

where we have defined $\mathbf{Z} \equiv \mathbf{\Sigma V}^T$. Note that the previous columns $\{\hat{\mathbf{u}}_\mathbf{1}, \hat{\mathbf{u}}_\mathbf{2}, \ldots, \hat{\mathbf{u}}_\mathbf{n}\}$ are now rows in $\mathbf{U}^T$. Comparing this equation to Equation 1, $\{\hat{\mathbf{u}}_\mathbf{1}, \hat{\mathbf{u}}_\mathbf{2}, \ldots, \hat{\mathbf{u}}_\mathbf{n}\}$ perform the same role as $\{\hat{\mathbf{p}}_\mathbf{1}, \hat{\mathbf{p}}_\mathbf{2}, \ldots, \hat{\mathbf{p}}_\mathbf{m}\}$. Hence, $\mathbf{U}^T$ is a *change of basis* from $\mathbf{X}$ to $\mathbf{Z}$. Just as before, we were transforming column vectors, we can again infer that we are

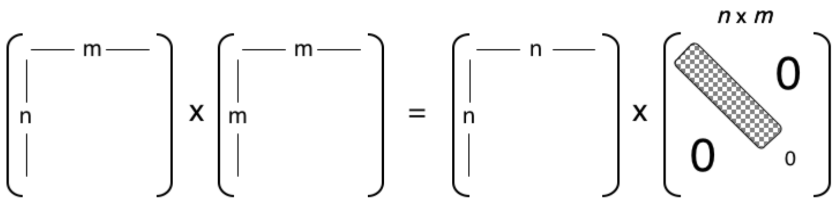
The "value" form of SVD is expressed in equation 7.

$$\mathbf{X}\hat{\mathbf{v}}_i = \sigma_i \hat{\mathbf{u}}_i$$

The mathematical intuition behind the construction of the matrix form is that we want to express all n "value" equations in just one equation. It is easiest to understand this process graphically. Drawing the matrices of equation 7 looks likes the following.



We can construct three new matrices $\mathbf{V}$, $\mathbf{U}$ and $\mathbf{\Sigma}$. All singular values are first rank-ordered $\sigma_{\tilde{1}} \geq \sigma_{\tilde{2}} \geq \ldots \geq \sigma_{\tilde{r}}$, and the corresponding vectors are indexed in the same rank order. Each pair of associated vectors $\hat{\mathbf{v}}_\mathbf{i}$ and $\hat{\mathbf{u}}_\mathbf{i}$ is stacked in the i^{th} column along their respective matrices. The corresponding singular value σ_i is placed along the diagonal (the ii^{th} position) of $\mathbf{\Sigma}$. This generates the equation $\mathbf{XV} = \mathbf{U\Sigma}$, which looks like the following.



The matrices $\mathbf{V}$ and $\mathbf{U}$ are $m \times m$ and $n \times n$ matrices respectively and $\mathbf{\Sigma}$ is a diagonal matrix with a few non-zero values (represented by the checkerboard) along its diagonal. Solving this single matrix equation solves all n "value" form equations.

Figure 4: How to construct the matrix form of SVD from the "value" form.

transforming column vectors. The fact that the orthonormal basis $\mathbf{U}^T$ (or $\mathbf{P}$) transforms column vectors means that $\mathbf{U}^T$ is a basis that spans the columns of $\mathbf{X}$. Bases that span the columns are termed the *column space* of $\mathbf{X}$. The column space formalizes the notion of what are the possible "outputs" of any matrix.

There is a funny symmetry to *SVD* such that we can define a similar quantity - the *row space*.

$$\begin{aligned} \mathbf{XV} &= \mathbf{\Sigma U} \\ (\mathbf{XV})^T &= (\mathbf{\Sigma U})^T \\ \mathbf{V}^T\mathbf{X}^T &= \mathbf{U}^T\mathbf{\Sigma} \\ \mathbf{V}^T\mathbf{X}^T &= \mathbf{Z} \end{aligned}$$

where we have defined $\mathbf{Z} \equiv \mathbf{U^T\Sigma}$. Again the rows of $\mathbf{V}^T$ (or the columns of $\mathbf{V}$) are an orthonormal basis for transforming $\mathbf{X}^T$ into $\mathbf{Z}$. Because of the transpose on $\mathbf{X}$, it follows that $\mathbf{V}$ is an orthonormal basis spanning the *row space* of $\mathbf{X}$. The row space likewise formalizes the notion of what are possible "inputs" into an arbitrary matrix.

We are only scratching the surface for understanding the full implications of *SVD*. For the purposes of this tutorial though, we have enough information to understand how *PCA* will fall within this framework.

6.3 SVD and PCA

With similar computations it is evident that the two methods are intimately related. Let us return to the original $m \times n$ data matrix $\mathbf{X}$. We can define a new matrix $\mathbf{Y}$ as an $n \times m$ matrix[9].

$$\mathbf{Y} \equiv \frac{1}{\sqrt{n-1}}\mathbf{X}^T$$

where each *column* of $\mathbf{Y}$ has zero mean. The definition of $\mathbf{Y}$ becomes clear by analyzing $\mathbf{Y}^T\mathbf{Y}$.

$$\begin{aligned}
\mathbf{Y}^T\mathbf{Y} &= \left(\frac{1}{\sqrt{n-1}}\mathbf{X}^T\right)^T\left(\frac{1}{\sqrt{n-1}}\mathbf{X}^T\right) \\
&= \frac{1}{n-1}\mathbf{X}^{TT}\mathbf{X}^T \\
&= \frac{1}{n-1}\mathbf{X}\mathbf{X}^T \\
\mathbf{Y}^T\mathbf{Y} &= \mathbf{S_X}
\end{aligned}$$

By construction $\mathbf{Y}^T\mathbf{Y}$ equals the covariance matrix of $\mathbf{X}$. From section 5 we know that the principal components of $\mathbf{X}$ are the eigenvectors of $\mathbf{S_X}$. If we calculate the *SVD* of $\mathbf{Y}$, the columns of matrix $\mathbf{V}$ contain the eigenvectors of $\mathbf{Y}^T\mathbf{Y} = \mathbf{S_X}$. *Therefore, the columns of* $\mathbf{V}$ *are the principal components of* $\mathbf{X}$. This second algorithm is encapsulated in Matlab code included in Appendix B.

What does this mean? $\mathbf{V}$ spans the row space of $\mathbf{Y} \equiv \frac{1}{\sqrt{n-1}}\mathbf{X}^T$. Therefore, $\mathbf{V}$ must also span the column space of $\frac{1}{\sqrt{n-1}}\mathbf{X}$. We can conclude that finding the principal components[10] amounts to finding an orthonormal basis that spans the *column space* of $\mathbf{X}$.

7 Discussion and Conclusions

7.1 Quick Summary

Performing *PCA* is quite simple in practice.

[9] $\mathbf{Y}$ is of the appropriate $n \times m$ dimensions laid out in the derivation of section 6.1. This is the reason for the "flipping" of dimensions in 6.1 and Figure 4.

[10] If the final goal is to find an orthonormal basis for the coulmn space of $\mathbf{X}$ then we can calculate it directly without constructing $\mathbf{Y}$. By symmetry the columns of $\mathbf{U}$ produced by the *SVD* of $\frac{1}{\sqrt{n-1}}\mathbf{X}$ must also be the principal components.

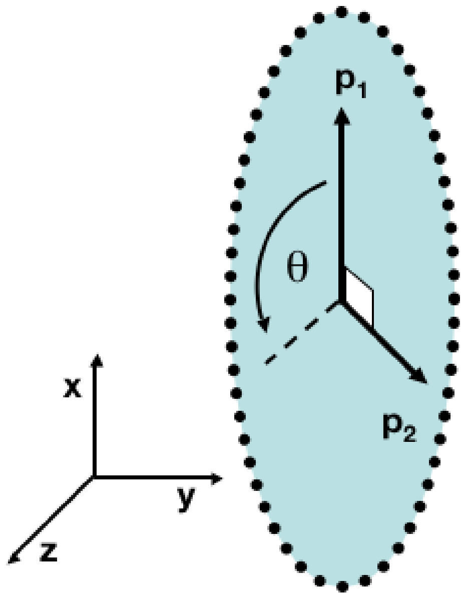


Figure 5: Data points (black dots) tracking a person on a ferris wheel. The extracted principal components are $(\mathbf{p_1}, \mathbf{p_2})$ and the phase is $\hat{\theta}$.

1. Organize a data set as an $m \times n$ matrix, where m is the number of measurement types and n is the number of trials.
2. Subtract off the mean for each measurement type or row $\mathbf{x_i}$.
3. Calculate the *SVD* or the eigenvectors of the covariance.

In several fields of literature, many authors refer to the individual measurement types $\mathbf{x_i}$ as the *sources*. The data projected into the principal components $\mathbf{Y} = \mathbf{PX}$ are termed the *signals*, because the projected data presumably represent the "true" underlying probability distributions.

7.2 Dimensional Reduction

One benefit of *PCA* is that we can examine the variances $\mathbf{S_Y}$ associated with the principle components. Often one finds that large variances associated with the first $k < m$ principal components, and then a precipitous drop-off. One can conclude that most interesting dynamics occur only in the first k dimensions.

In the example of the spring, $k = 1$. Likewise, in Figure 3 panel (c), we recognize $k = 1$ along the principal component of $r_2 = kr_1$. This process of of throwing out the less important axes can help reveal hidden, simplified dynamics in high dimensional data. This process is aptly named *dimensional reduction*.

7.3 Limits and Extensions of PCA

Both the strength and weakness of *PCA* is that it is a *non-parametric* analysis. One only needs to make the assumptions outlined in section 4.5 and then calculate the corresponding answer. There are no parameters to tweak and no coefficients to adjust based on user experience - the answer is unique[11] and independent of the user.

This same strength can also be viewed as a weakness. If one knows a-priori some features of the dynamics of a system, then it makes sense to incorporate these assumptions into a *parametric* algorithm - or an algorithm with selected parameters.

Consider the recorded positions of a person on a ferris wheel over time in Figure 5. The probability distributions along the axes are approximately Gaussian and thus *PCA* finds $(\mathbf{p_1}, \mathbf{p_2})$, however this answer might not be optimal. The most concise form of dimensional reduction is to recognize that the phase (or angle along the ferris wheel) contains all dynamic information. Thus, the appropriate parametric algorithm is to first convert the data to the appropriately centered polar coordinates and then compute *PCA*.

This prior *non-linear* transformation is sometimes termed a *kernel transformation* and the entire parametric algorithm is termed *kernel PCA*. Other common kernel transformations include Fourier and Gaussian transformations. This procedure is parametric because the user must incorporate prior knowledge of the dynamics in the selection of the kernel but it is also more optimal in the sense that the dynamics are more concisely described.

[11] To be absolutely precise, the principal components are **not** uniquely defined. One can always flip the direction by multiplying by -1. In addition, eigenvectors beyond the rank of a matrix (i.e. $\sigma_i = 0$ for $i > rank$) can be selected almost at whim. However, these degrees of freedom do not effect the qualitative features of the solution nor a dimensional reduction.

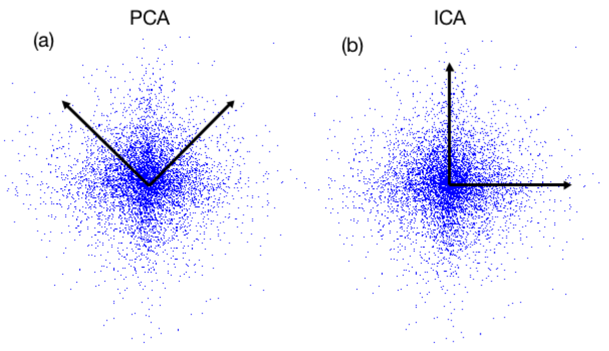


Figure 6: Non-Gaussian distributed data causes *PCA* to fail. In exponentially distributed data the axes with the largest variance do not correspond to the underlying basis.

Sometimes though the assumptions themselves are too stringent. One might envision situations where the principal components need not be orthogonal. Furthermore, the distributions along each dimension $(\mathbf{x_i})$ need not be Gaussian. For instance, Figure 6 contains a 2-D exponentially distributed data set. The largest variances do not correspond to the meaningful axes of thus *PCA* fails.

This less constrained set of problems is not trivial and only recently has been solved adequately via *Independent Component Analysis (ICA)*. The formulation is equivalent.

> Find a matrix $\mathbf{P}$ where $\mathbf{Y} = \mathbf{PX}$ such that $\mathbf{S_Y}$ is diagonalized.

however it abandons all assumptions except linearity, and attempts to find axes that satisfy the most formal form of redundancy reduction - *statistical independence*. Mathematically *ICA* finds a basis such that the joint probability distribution can be factorized[12].

$$P(\mathbf{y_i}, \mathbf{y_j}) = P(\mathbf{y_i})P(\mathbf{y_j})$$

for all i and j, $i \neq j$. The downside of *ICA* is that it is a form of nonlinear optimizaiton, making the solution difficult to calculate in practice and potentially not unique. However *ICA* has been shown a very practical and powerful algorithm for solving a whole new class of problems.

Writing this paper has been an extremely instructional experience for me. I hope that this paper helps to demystify the motivation and results of *PCA*, and the underlying assumptions behind this important analysis technique.

---

8 Appendix A: Linear Algebra

This section proves a few unapparent theorems in linear algebra, which are crucial to this paper.

1. The inverse of an orthogonal matrix is its transpose.

The goal of this proof is to show that if $\mathbf{A}$ is an orthogonal matrix, then $\mathbf{A}^{-1} = \mathbf{A}^T$.

Let $\mathbf{A}$ be an $m \times n$ matrix.

$$\mathbf{A} = [\mathbf{a_1}\ \mathbf{a_2}\ \ldots\ \mathbf{a_n}]$$

where $\mathbf{a_i}$ is the i^{th} *column* vector. We now show that $\mathbf{A}^T\mathbf{A} = \mathbf{I}$ where $\mathbf{I}$ is the identity matrix.

Let us examine the ij^{th} element of the matrix $\mathbf{A}^T\mathbf{A}$. The ij^{th} element of $\mathbf{A}^T\mathbf{A}$ is $(\mathbf{A}^T\mathbf{A})_{ij} = \mathbf{a_i}^T\mathbf{a_j}$.

Remember that the columns of an orthonormal matrix are orthonormal to each other. In other words, the dot product of any two columns is zero. The only exception is a dot product of one particular column with itself, which equals one.

$$(\mathbf{A}^T\mathbf{A})_{ij} = \mathbf{a_i}^T\mathbf{a_j} = \left\{ \begin{array}{ll} 1 & i = j \\ 0 & i \neq j \end{array} \right.$$

$\mathbf{A}^T\mathbf{A}$ is the exact description of the identity matrix. The definition of $\mathbf{A}^{-1}$ is $\mathbf{A}^{-1}\mathbf{A} = \mathbf{I}$. Therefore, because $\mathbf{A}^T\mathbf{A} = \mathbf{I}$, it follows that $\mathbf{A}^{-1} = \mathbf{A}^T$.

[12]Equivalently, in the language of *information theory* the goal is to find a basis $\mathbf{P}$ such that the mutual information $I(\mathbf{y_i}, \mathbf{y_j}) = 0$ for $i \neq j$.

2. If A is any matrix, the matrices $\mathbf{A}^T\mathbf{A}$ and $\mathbf{A}\mathbf{A}^T$ are both symmetric.

Let's examine the transpose of each in turn.

$$\begin{aligned} (\mathbf{A}\mathbf{A}^T)^T &= \mathbf{A}^{TT}\mathbf{A}^T = \mathbf{A}\mathbf{A}^T \\ (\mathbf{A}^T\mathbf{A})^T &= \mathbf{A}^T\mathbf{A}^{TT} = \mathbf{A}^T\mathbf{A} \end{aligned}$$

The equality of the quantity with its transpose completes this proof.

3. A matrix is symmetric if and only if it is orthogonally diagonalizable.

Because this statement is bi-directional, it requires a two-part "if-and-only-if" proof. One needs to prove the forward and the backwards "if-then" cases.

Let us start with the forward case. If $\mathbf{A}$ is orthogonally diagonalizable, then $\mathbf{A}$ is a symmetric matrix. By hypothesis, orthogonally diagonalizable means that there exists some $\mathbf{E}$ such that $\mathbf{A} = \mathbf{E}\mathbf{D}\mathbf{E}^T$, where $\mathbf{D}$ is a diagonal matrix and $\mathbf{E}$ is some special matrix which diagonalizes $\mathbf{A}$. Let us compute $\mathbf{A}^T$.

$$\mathbf{A}^T = (\mathbf{E}\mathbf{D}\mathbf{E}^T)^T = \mathbf{E}^{TT}\mathbf{D}^T\mathbf{E}^T = \mathbf{E}\mathbf{D}\mathbf{E}^T = \mathbf{A}$$

Evidently, if $\mathbf{A}$ is orthogonally diagonalizable, it must also be symmetric.

The reverse case is more involved and less clean so it will be left to the reader. In lieu of this, hopefully the "forward" case is suggestive if not somewhat convincing.

4. A symmetric matrix is diagonalized by a matrix of its orthonormal eigenvectors.

Restated in math, let $\mathbf{A}$ be a square $n \times n$ symmetric matrix with associated eigenvectors $\{\mathbf{e_1}, \mathbf{e_2}, \ldots, \mathbf{e_n}\}$. Let $\mathbf{E} = [\mathbf{e_1}\ \mathbf{e_2}\ \ldots\ \mathbf{e_n}]$ where the i^{th} column of $\mathbf{E}$ is the eigenvector $\mathbf{e_i}$. This theorem asserts that there exists a diagonal matrix D where $\mathbf{A} = \mathbf{E}\mathbf{D}\mathbf{E}^T$.

This theorem is an extension of the previous theorem 3. It provides a prescription for how to find the matrix $\mathbf{E}$, the "diagonalizer" for a symmetric matrix. It says that the special diagonalizer is in fact a matrix of the original matrix's eigenvectors.

This proof is in two parts. In the first part, we see that the any matrix can be orthogonally diagonalized if and only if it that matrix's eigenvectors are all linearly independent. In the second part of the proof, we see that a symmetric matrix has the special property that all of its eigenvectors are not just linearly independent but also orthogonal, thus completing our proof.

In the first part of the proof, let $\mathbf{A}$ be just some matrix, not necessarily symmetric, and let it have independent eigenvectors (i.e. no degeneracy). Furthermore, let $\mathbf{E} = [\mathbf{e_1}\ \mathbf{e_2}\ \dots\ \mathbf{e_n}]$ be the matrix of eigenvectors placed in the columns. Let $\mathbf{D}$ be a diagonal matrix where the i^{th} eigenvalue is placed in the ii^{th} position.

We will now show that $\mathbf{AE} = \mathbf{ED}$. We can examine the columns of the right-hand and left-hand sides of the equation.

$$\begin{array}{rrcl} \textsf{Left hand side :} & \mathbf{AE} & = & [\mathbf{Ae_1}\ \mathbf{Ae_2}\ \dots\ \mathbf{Ae_n}] \\ \textsf{Right hand side :} & \mathbf{ED} & = & [\lambda_1\mathbf{e_1}\ \lambda_2\mathbf{e_2}\ \dots\ \lambda_n\mathbf{e_n}] \end{array}$$

Evidently, if $\mathbf{AE} = \mathbf{ED}$ then $\mathbf{Ae_i} = \lambda_i\mathbf{e_i}$ for all i. This equation is the definition of the eigenvalue equation. Therefore, it must be that $\mathbf{AE} = \mathbf{ED}$. A little rearrangement provides $\mathbf{A} = \mathbf{EDE}^{-1}$, completing the first part the proof.

For the second part of the proof, we show that a symmetric matrix always has orthogonal eigenvectors. For some symmetric matrix, let λ_1 and λ_2 be distinct eigenvalues for eigenvectors $\mathbf{e_1}$ and $\mathbf{e_2}$.

$$\begin{array}{rcl} \lambda_1\mathbf{e_1}\cdot\mathbf{e_2} & = & (\lambda_1\mathbf{e_1})^T\mathbf{e_2} \\ & = & (\mathbf{Ae_1})^T\mathbf{e_2} \\ & = & \mathbf{e_1}^T\mathbf{A}^T\mathbf{e_2} \\ & = & \mathbf{e_1}^T\mathbf{Ae_2} \\ & = & \mathbf{e_1}^T(\lambda_2\mathbf{e_2}) \\ \lambda_1\mathbf{e_1}\cdot\mathbf{e_2} & = & \lambda_2\mathbf{e_1}\cdot\mathbf{e_2} \end{array}$$

By the last relation we can equate that $(\lambda_1 - \lambda_2)\mathbf{e_1}\cdot\mathbf{e_2} = 0$. Since we have conjectured that the eigenvalues are in fact unique, it must be the case that $\mathbf{e_1}\cdot\mathbf{e_2} = 0$. Therefore, the eigenvectors of a symmetric matrix are orthogonal.

Let us back up now to our original postulate that $\mathbf{A}$ is a symmetric matrix. By the second part of the proof, we know that the eigenvectors of $\mathbf{A}$ are all orthonormal (we choose the eigenvectors to be normalized). This means that $\mathbf{E}$ is an orthogonal matrix so by theorem 1, $\mathbf{E}^T = \mathbf{E}^{-1}$ and we can rewrite the final result.

$$\mathbf{A} = \mathbf{EDE}^T$$

. Thus, a symmetric matrix is diagonalized by a matrix of its eigenvectors.

5. For any arbitrary $m \times n$ matrix X, the symmetric matrix $\mathbf{X}^T\mathbf{X}$ has a set of orthonormal eigenvectors of $\{\hat{\mathbf{v}}_1, \hat{\mathbf{v}}_2, \dots, \hat{\mathbf{v}}_n\}$ and a set of associated eigenvalues $\{\lambda_1, \lambda_2, \dots, \lambda_n\}$. The set of vectors $\{\mathbf{X}\hat{\mathbf{v}}_1, \mathbf{X}\hat{\mathbf{v}}_2, \dots, \mathbf{X}\hat{\mathbf{v}}_n\}$ then form an orthogonal basis, where each vector $\mathbf{X}\hat{\mathbf{v}}_i$ is of length $\sqrt{\lambda_i}$.

All of these properties arise from the dot product of any two vectors from this set.

$$\begin{array}{rcl} (\mathbf{X}\hat{\mathbf{v}}_i)\cdot(\mathbf{X}\hat{\mathbf{v}}_j) & = & (\mathbf{X}\hat{\mathbf{v}}_i)^T(\mathbf{X}\hat{\mathbf{v}}_j) \\ & = & \hat{\mathbf{v}}_i^T\mathbf{X}^T\mathbf{X}\hat{\mathbf{v}}_j \\ & = & \hat{\mathbf{v}}_i^T(\lambda_j\hat{\mathbf{v}}_j) \\ & = & \lambda_j\hat{\mathbf{v}}_i\cdot\hat{\mathbf{v}}_j \\ (\mathbf{X}\hat{\mathbf{v}}_i)\cdot(\mathbf{X}\hat{\mathbf{v}}_j) & = & \lambda_j\delta_{ij} \end{array}$$

The last relation arises because the set of eigenvectors of $\mathbf{X}$ is orthogonal resulting in the Kronecker delta. In more simpler terms the last relation states:

$$(\mathbf{X}\hat{\mathbf{v}}_i)\cdot(\mathbf{X}\hat{\mathbf{v}}_j) = \left\{ \begin{array}{ll} \lambda_j & i = j \\ 0 & i \neq j \end{array} \right.$$

This equation states that any two vectors in the set are orthogonal.

The second property arises from the above equation by realizing that the length squared of each vector is defined as:

$$\|\mathbf{X}\hat{\mathbf{v}}_i\|^2 = (\mathbf{X}\hat{\mathbf{v}}_i)\cdot(\mathbf{X}\hat{\mathbf{v}}_i) = \lambda_i$$

9 Appendix B: Code

This code is written for Matlab 6.5 (Release 13) from Mathworks[13]. The code is not computationally efficient but explanatory (terse comments begin with a %).

This first version follows Section 5 by examining the covariance of the data set.

```
function [signals,PC,V] = pca1(data)
% PCA1: Perform PCA using covariance.
%     data - MxN matrix of input data
%            (M dimensions, N trials)
%  signals - MxN matrix of projected data
%       PC - each column is a PC
%        V - Mx1 matrix of variances

[M,N] = size(data);

% subtract off the mean for each dimension
mn =  mean(data,2);
data = data - repmat(mn,1,N);

% calculate the covariance matrix
covariance = 1 / (N-1) * data * data';

% find the eigenvectors and eigenvalues
[PC, V] = eig(covariance);

% extract diagonal of matrix as vector
V = diag(V);

% sort the variances in decreasing order
[junk, rindices] = sort(-1*V);
V  = V(rindices);
PC = PC(:,rindices);

% project the original data set
signals = PC' * data;
```

This second version follows section 6 computing *PCA* through *SVD*.

```
function [signals,PC,V] = pca2(data)
% PCA2: Perform PCA using SVD.
%     data - MxN matrix of input data
%            (M dimensions, N trials)
%  signals - MxN matrix of projected data
%       PC - each column is a PC
%        V - Mx1 matrix of variances

[M,N] = size(data);

% subtract off the mean for each dimension
mn =  mean(data,2);
data = data - repmat(mn,1,N);

% construct the matrix Y
Y = data' / sqrt(N-1);

% SVD does it all
[u,S,PC] = svd(Y);

% calculate the variances
S = diag(S);
V = S .* S;

% project the original data
signals = PC' * data;
```

[13] http://www.mathworks.com

10 References

Bell, Anthony and Sejnowski, Terry. (1997) “The Independent Components of Natural Scenes are Edge Filters.” *Vision Research* 37(23), 3327-3338.
[A paper from my field of research that surveys and explores different forms of decorrelating data sets. The authors examine the features of PCA and compare it with new ideas in redundancy reduction, namely Independent Component Analysis.]

Bishop, Christopher. (1996) *Neural Networks for Pattern Recognition*. Clarendon, Oxford, UK.
[A challenging but brilliant text on statistical pattern recognition (neural networks). Although the derivation of PCA is touch in section 8.6 (p.310-319), it does have a great discussion on potential extensions to the method and it puts PCA in context of other methods of dimensional

reduction. Also, I want to acknowledge this book for several ideas about the limitations of PCA.]

Lay, David. (2000). *Linear Algebra and It's Applications*. Addison-Wesley, New York.
[This is a beautiful text. Chapter 7 in the second edition (p. 441-486) has an exquisite, intuitive derivation and discussion of SVD and PCA. Extremely easy to follow and a must read.]

Mitra, Partha and Pesaran, Bijan. (1999) "Analysis of Dynamic Brain Imaging Data." *Biophysical Journal*. 76, 691-708.
[A comprehensive and spectacular paper from my field of research interest. It is dense but in two sections "Eigenmode Analysis: SVD" and "Space-frequency SVD" the authors discuss the benefits of performing a Fourier transform on the data before an SVD.]

Will, Todd (1999) "Introduction to the Singular Value Decomposition" Davidson College. *www.davidson.edu/academic/math/will/svd/index.html*
[A math professor wrote up a great web tutorial on SVD with tremendous intuitive explanations, graphics and animations. Although it avoids PCA directly, it gives a great intuitive feel for what SVD is doing mathematically. Also, it is the inspiration for my "spring" example.]

A tutorial on Principal Components Analysis

Lindsay I Smith

February 26, 2002

Chapter 1

Introduction

This tutorial is designed to give the reader an understanding of Principal Components Analysis (PCA). PCA is a useful statistical technique that has found application in fields such as face recognition and image compression, and is a common technique for finding patterns in data of high dimension.

Before getting to a description of PCA, this tutorial first introduces mathematical concepts that will be used in PCA. It covers standard deviation, covariance, eigenvectors and eigenvalues. This background knowledge is meant to make the PCA section very straightforward, but can be skipped if the concepts are already familiar.

There are examples all the way through this tutorial that are meant to illustrate the concepts being discussed. If further information is required, the mathematics textbook "Elementary Linear Algebra 5e" by Howard Anton, Publisher John Wiley & Sons Inc, ISBN 0-471-85223-6 is a good source of information regarding the mathematical background.

Chapter 2

Background Mathematics

This section will attempt to give some elementary background mathematical skills that will be required to understand the process of Principal Components Analysis. The topics are covered independently of each other, and examples given. It is less important to remember the exact mechanics of a mathematical technique than it is to understand the reason why such a technique may be used, and what the result of the operation tells us about our data. Not all of these techniques are used in PCA, but the ones that are not explicitly required do provide the grounding on which the most important techniques are based.

I have included a section on Statistics which looks at distribution measurements, or, how the data is spread out. The other section is on Matrix Algebra and looks at eigenvectors and eigenvalues, important properties of matrices that are fundamental to PCA.

2.1 Statistics

The entire subject of statistics is based around the idea that you have this big set of data, and you want to analyse that set in terms of the relationships between the individual points in that data set. I am going to look at a few of the measures you can do on a set of data, and what they tell you about the data itself.

2.1.1 Standard Deviation

To understand standard deviation, we need a data set. Statisticians are usually concerned with taking a *sample* of a *population*. To use election polls as an example, the population is all the people in the country, whereas a sample is a subset of the population that the statisticians measure. The great thing about statistics is that by only measuring (in this case by doing a phone survey or similar) a sample of the population, you can work out what is most likely to be the measurement if you used the entire population. In this statistics section, I am going to assume that our data sets are samples of some bigger population. There is a reference later in this section pointing to more information about samples and populations.

Here's an example set:

$$X = [\ 1\ 2\ 4\ 6\ 12\ 15\ 25\ 45\ 68\ 67\ 65\ 98\ ]$$

I could simply use the symbol X to refer to this entire set of numbers. If I want to refer to an individual number in this data set, I will use subscripts on the symbol X to indicate a specific number. Eg. X_3 refers to the 3rd number in X, namely the number 4. Note that X_1 is the first number in the sequence, not X_0 like you may see in some textbooks. Also, the symbol n will be used to refer to the number of elements in the set X

There are a number of things that we can calculate about a data set. For example, we can calculate the mean of the sample. I assume that the reader understands what the mean of a sample is, and will only give the formula:

$$\bar{X} = \frac{\sum_{i=1}^{n} X_i}{n}$$

Notice the symbol $\bar{X}$ (said "X bar") to indicate the mean of the set X. All this formula says is "Add up all the numbers and then divide by how many there are".

Unfortunately, the mean doesn't tell us a lot about the data except for a sort of middle point. For example, these two data sets have exactly the same mean (10), but are obviously quite different:

$$[\ 0\ 8\ 12\ 20\ ]\ and\ [\ 8\ 9\ 11\ 12\ ]$$

So what is different about these two sets? It is the *spread* of the data that is different. The Standard Deviation (SD) of a data set is a measure of how spread out the data is.

How do we calculate it? The English definition of the SD is: "The average distance from the mean of the data set to a point". The way to calculate it is to compute the squares of the distance from each data point to the mean of the set, add them all up, divide by $n-1$, and take the positive square root. As a formula:

$$s = \sqrt{\frac{\sum_{i=1}^{n}(X_i - \bar{X})^2}{(n-1)}}$$

Where s is the usual symbol for standard deviation of a sample. I hear you asking "Why are you using $(n-1)$ and not n?". Well, the answer is a bit complicated, but in general, if your data set is a *sample* data set, ie. you have taken a subset of the real-world (like surveying 500 people about the election) then you must use $(n-1)$ because it turns out that this gives you an answer that is closer to the standard deviation that would result if you had used the *entire* population, than if you'd used n. If, however, you are not calculating the standard deviation for a sample, but for an entire population, then you should divide by n instead of $(n-1)$. For further reading on this topic, the web page *http://mathcentral.uregina.ca/RR/database/RR.09.95/weston2.html* describes standard deviation in a similar way, and also provides an example experiment that shows the

Set 1:

X	$(X - \bar{X})$	$(X - \bar{X})^2$
0	-10	100
8	-2	4
12	2	4
20	10	100
Total		208
Divided by (n-1)		69.333
Square Root		8.3266

Set 2:

X_i	$(X_i - \bar{X})$	$(X_i - \bar{X})^2$
8	-2	4
9	-1	1
11	1	1
12	2	4
Total		10
Divided by (n-1)		3.333
Square Root		1.8257

Table 2.1: Calculation of standard deviation

difference between each of the denominators. It also discusses the difference between samples and populations.

So, for our two data sets above, the calculations of standard deviation are in Table 2.1.

And so, as expected, the first set has a much larger standard deviation due to the fact that the data is much more spread out from the mean. Just as another example, the data set:

$$[\ 10\ 10\ 10\ 10\ ]$$

also has a mean of 10, but its standard deviation is 0, because all the numbers are the same. None of them deviate from the mean.

2.1.2 Variance

Variance is another measure of the spread of data in a data set. In fact it is almost identical to the standard deviation. The formula is this:

$$s^2 = \frac{\sum_{i=1}^{n}(X_i - \bar{X})^2}{(n-1)}$$

You will notice that this is simply the standard deviation squared, in both the symbol (s^2) and the formula (there is no square root in the formula for variance). s^2 is the usual symbol for variance of a sample. Both these measurements are measures of the spread of the data. Standard deviation is the most common measure, but variance is also used. The reason why I have introduced variance in addition to standard deviation is to provide a solid platform from which the next section, covariance, can launch from.

Exercises

Find the mean, standard deviation, and variance for each of these data sets.

- [12 23 34 44 59 70 98]
- [12 15 25 27 32 88 99]
- [15 35 78 82 90 95 97]

2.1.3 Covariance

The last two measures we have looked at are purely 1-dimensional. Data sets like this could be: heights of all the people in the room, marks for the last COMP101 exam etc. However many data sets have more than one dimension, and the aim of the statistical analysis of these data sets is usually to see if there is any relationship between the dimensions. For example, we might have as our data set both the height of all the students in a class, and the mark they received for that paper. We could then perform statistical analysis to see if the height of a student has any effect on their mark.

Standard deviation and variance only operate on 1 dimension, so that you could only calculate the standard deviation for each dimension of the data set *independently* of the other dimensions. However, it is useful to have a similar measure to find out how much the dimensions vary from the mean *with respect to each other*.

Covariance is such a measure. Covariance is always measured *between* 2 dimensions. If you calculate the covariance between one dimension and *itself*, you get the variance. So, if you had a 3-dimensional data set (x, y, z), then you could measure the covariance between the x and y dimensions, the x and z dimensions, and the y and z dimensions. Measuring the covariance between x and x, or y and y, or z and z would give you the variance of the x, y and z dimensions respectively.

The formula for covariance is very similar to the formula for variance. The formula for variance could also be written like this:

$$var(X) = \frac{\sum_{i=1}^{n}(X_i - \bar{X})(X_i - \bar{X})}{(n-1)}$$

where I have simply expanded the square term to show both parts. So given that knowledge, here is the formula for covariance:

$$cov(X,Y) = \frac{\sum_{i=1}^{n}(X_i - \bar{X})(Y_i - \bar{Y})}{(n-1)}$$

includegraphicscovPlot.ps

Figure 2.1: A plot of the covariance data showing positive relationship between the number of hours studied against the mark received

It is exactly the same except that in the second set of brackets, the X's are replaced by Y's. This says, in English, "For each data item, multiply the difference between the x value and the mean of x, by the the difference between the y value and the mean of y. Add all these up, and divide by $(n-1)$".

How does this work? Lets use some example data. Imagine we have gone into the world and collected some 2-dimensional data, say, we have asked a bunch of students how many hours in total that they spent studying COSC241, and the mark that they received. So we have two dimensions, the first is the H dimension, the hours studied, and the second is the M dimension, the mark received. Figure 2.2 holds my imaginary data, and the calculation of $cov(H, M)$, the covariance between the Hours of study done and the Mark received.

So what does it tell us? The exact value is not as important as it's sign (ie. positive or negative). If the value is positive, as it is here, then that indicates that both dimensions *increase together*, meaning that, in general, as the number of hours of study increased, so did the final mark.

If the value is negative, then as one dimension increases, the other decreases. If we had ended up with a negative covariance here, then that would have said the opposite, that as the number of hours of study increased the the final mark *decreased*.

In the last case, if the covariance is zero, it indicates that the two dimensions are independent of each other.

The result that mark given increases as the number of hours studied increases can be easily seen by drawing a graph of the data, as in Figure 2.1.3. However, the luxury of being able to visualize data is only available at 2 and 3 dimensions. Since the covariance value can be calculated between any 2 dimensions in a data set, this technique is often used to find relationships between dimensions in high-dimensional data sets where visualisation is difficult.

You might ask "is $cov(X, Y)$ equal to $cov(Y, X)$"? Well, a quick look at the formula for covariance tells us that yes, they are exactly the same since the only difference between $cov(X, Y)$ and $cov(Y, X)$ is that $(X_i - \bar{X})(Y_i - \bar{Y})$ is replaced by $(Y_i - \bar{Y})(X_i - \bar{X})$. And since multiplication is commutative, which means that it doesn't matter which way around I multiply two numbers, I always get the same number, these two equations give the same answer.

2.1.4 The covariance Matrix

Recall that covariance is always measured between 2 dimensions. If we have a data set with more than 2 dimensions, there is more than one covariance measurement that can be calculated. For example, from a 3 dimensional data set (dimensions x, y, z) you could calculate $cov(x, y)$, $(cov(x, z)$, and $cov(y, z)$. In fact, for an n-dimensional data set, you can calculate $\frac{n!}{(n-2)!*2}$ different covariance values.

	Hours(H)	*Mark(M)*
Data	9	39
	15	56
	25	93
	14	61
	10	50
	18	75
	0	32
	16	85
	5	42
	19	70
	16	66
	20	80
Totals	167	749
Averages	13.92	62.42

Covariance:

H	M	$(H_i - \bar{H})$	$(M_i - \bar{M})$	$(H_i - \bar{H})(M_i - \bar{M})$
9	39	-4.92	-23.42	115.23
15	56	1.08	-6.42	-6.93
25	93	11.08	30.58	338.83
14	61	0.08	-1.42	-0.11
10	50	-3.92	-12.42	48.69
18	75	4.08	12.58	51.33
0	32	-13.92	-30.42	423.45
16	85	2.08	22.58	46.97
5	42	-8.92	-20.42	182.15
19	70	5.08	7.58	38.51
16	66	2.08	3.58	7.45
20	80	6.08	17.58	106.89
Total				1149.89
Average				104.54

Table 2.2: 2-dimensional data set and covariance calculation

A useful way to get all the possible covariance values between all the different dimensions is to calculate them all and put them in a matrix. I assume in this tutorial that you are familiar with matrices, and how they can be defined. So, the definition for the covariance matrix for a set of data with n dimensions is:

$$C^{n\times n} = (c_{i,j},\ c_{i,j} = cov(Dim_i, Dim_j)),$$

where $C^{n\times n}$ is a matrix with n rows and n columns, and Dim_x is the xth dimension. All that this ugly looking formula says is that if you have an n-dimensional data set, then the matrix has n rows and columns (so is square) and each entry in the matrix is the result of calculating the covariance between two separate dimensions. Eg. the entry on row 2, column 3, is the covariance value calculated between the 2nd dimension and the 3rd dimension.

An example. We'll make up the covariance matrix for an imaginary 3 dimensional data set, using the usual dimensions x, y and z. Then, the covariance matrix has 3 rows and 3 columns, and the values are this:

$$C = \begin{pmatrix} cov(x,x) & cov(x,y) & cov(x,z) \\ cov(y,x) & cov(y,y) & cov(y,z) \\ cov(z,x) & cov(z,y) & cov(z,z) \end{pmatrix}$$

Some points to note: Down the main diagonal, you see that the covariance value is between one of the dimensions and itself. These are the variances for that dimension. The other point is that since $cov(a,b) = cov(b,a)$, the matrix is symmetrical about the main diagonal.

Exercises

Work out the covariance between the x and y dimensions in the following 2 dimensional data set, and describe what the result indicates about the data.

Item Number:	1	2	3	4	5
x	10	39	19	23	28
y	43	13	32	21	20

Calculate the covariance matrix for this 3 dimensional set of data.

Item Number:	1	2	3
x	1	-1	4
y	2	1	3
z	1	3	-1

2.2 Matrix Algebra

This section serves to provide a background for the matrix algebra required in PCA. Specifically I will be looking at eigenvectors and eigenvalues of a given matrix. Again, I assume a basic knowledge of matrices.

$$\begin{pmatrix} 2 & 3 \\ 2 & 1 \end{pmatrix} \times \begin{pmatrix} 1 \\ 3 \end{pmatrix} = \begin{pmatrix} 11 \\ 5 \end{pmatrix}$$

$$\begin{pmatrix} 2 & 3 \\ 2 & 1 \end{pmatrix} \times \begin{pmatrix} 3 \\ 2 \end{pmatrix} = \begin{pmatrix} 12 \\ 8 \end{pmatrix} = 4 \times \begin{pmatrix} 3 \\ 2 \end{pmatrix}$$

Figure 2.2: Example of one non-eigenvector and one eigenvector

$$2 \times \begin{pmatrix} 3 \\ 2 \end{pmatrix} = \begin{pmatrix} 6 \\ 4 \end{pmatrix}$$

$$\begin{pmatrix} 2 & 3 \\ 2 & 1 \end{pmatrix} \times \begin{pmatrix} 6 \\ 4 \end{pmatrix} = \begin{pmatrix} 24 \\ 16 \end{pmatrix} = 4 \times \begin{pmatrix} 6 \\ 4 \end{pmatrix}$$

Figure 2.3: Example of how a scaled eigenvector is still and eigenvector

2.2.1 Eigenvectors

As you know, you can multiply two matrices together, provided they are compatible sizes. Eigenvectors are a special case of this. Consider the two multiplications between a matrix and a vector in Figure 2.2.

In the first example, the resulting vector is not an integer multiple of the original vector, whereas in the second example, the example is exactly 4 times the vector we began with. Why is this? Well, the vector is a vector in 2 dimensional space. The vector $\begin{pmatrix} 3 \\ 2 \end{pmatrix}$ (from the second example multiplication) represents an arrow pointing from the origin, $(0, 0)$, to the point $(3, 2)$. The other matrix, the square one, can be thought of as a transformation matrix. If you multiply this matrix on the left of a vector, the answer is another vector that is transformed from it's original position.

It is the nature of the transformation that the eigenvectors arise from. Imagine a transformation matrix that, when multiplied on the left, reflected vectors in the line $y = x$. Then you can see that if there were a vector that lay *on* the line $y = x$, it's reflection it *itself*. This vector (and all multiples of it, because it wouldn't matter how long the vector was), would be an eigenvector of that transformation matrix.

What properties do these eigenvectors have? You should first know that eigenvectors can only be found for *square* matrices. And, not every square matrix has eigenvectors. And, given an $n \times n$ matrix that does have eigenvectors, there are n of them. Given a 3×3 matrix, there are 3 eigenvectors.

Another property of eigenvectors is that even if I scale the vector by some amount before I multiply it, I still get the same multiple of it as a result, as in Figure 2.3. This is because if you scale a vector by some amount, all you are doing is making it longer, not changing it's direction. Lastly, all the eigenvectors of a matrix are *perpendicular*, ie. at right angles to each other, no matter how many dimensions you have. By the way, another word for perpendicular, in maths talk, is *orthogonal*. This is important because it means that you can express the data in terms of these perpendicular eigenvectors, instead of expressing them in terms of the x and y axes. We will be doing this later in the section on PCA.

Another important thing to know is that when mathematicians find eigenvectors, they like to find the eigenvectors whose length is exactly one. This is because, as you know, the length of a vector doesn't affect whether it's an eigenvector or not, whereas the direction does. So, in order to keep eigenvectors standard, whenever we find an eigenvector we usually scale it to make it have a length of 1, so that all eigenvectors have the same length. Here's a demonstration from our example above.

$$\begin{pmatrix} 3 \\ 2 \end{pmatrix}$$

is an eigenvector, and the length of that vector is

$$\sqrt{}(3^2 + 2^2) = \sqrt{}13$$

so we divide the original vector by this much to make it have a length of 1.

$$\begin{pmatrix} 3 \\ 2 \end{pmatrix} \div \sqrt{}13 = \begin{pmatrix} 3/\sqrt{}13 \\ 2/\sqrt{}13 \end{pmatrix}$$

How does one go about finding these mystical eigenvectors? Unfortunately, it's only easy(ish) if you have a rather small matrix, like no bigger than about 3×3. After that, the usual way to find the eigenvectors is by some complicated iterative method which is beyond the scope of this tutorial (and this author). If you ever need to find the eigenvectors of a matrix in a program, just find a maths library that does it all for you. A useful maths package, called newmat, is available at *http://webnz.com/robert/* .

Further information about eigenvectors in general, how to find them, and orthogonality, can be found in the textbook "Elementary Linear Algebra 5e" by Howard Anton, Publisher John Wiley & Sons Inc, ISBN 0-471-85223-6.

2.2.2 Eigenvalues

Eigenvalues are closely related to eigenvectors, in fact, we saw an eigenvalue in Figure 2.2. Notice how, in both those examples, the amount by which the original vector was scaled after multiplication by the square matrix was the same? In that example, the value was 4. 4 is the *eigenvalue* associated with that eigenvector. No matter what multiple of the eigenvector we took before we multiplied it by the square matrix, we would always get 4 times the scaled vector as our result (as in Figure 2.3).

So you can see that eigenvectors and eigenvalues always come in pairs. When you get a fancy programming library to calculate your eigenvectors for you, you usually get the eigenvalues as well.

Exercises

For the following square matrix:

$$\begin{pmatrix} 3 & 0 & 1 \\ -4 & 1 & 2 \\ -6 & 0 & -2 \end{pmatrix}$$

Decide which, if any, of the following vectors are eigenvectors of that matrix and give the corresponding eigenvalue.

$$\begin{pmatrix} 2 \\ 2 \\ -1 \end{pmatrix} \begin{pmatrix} -1 \\ 0 \\ 2 \end{pmatrix} \begin{pmatrix} -1 \\ 1 \\ 3 \end{pmatrix} \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} \begin{pmatrix} 3 \\ 2 \\ 1 \end{pmatrix}$$

Chapter 3

Principal Components Analysis

Finally we come to Principal Components Analysis (PCA). What is it? It is a way of identifying patterns in data, and expressing the data in such a way as to highlight their similarities and differences. Since patterns in data can be hard to find in data of high dimension, where the luxury of graphical representation is not available, PCA is a powerful tool for analysing data.

The other main advantage of PCA is that once you have found these patterns in the data, and you compress the data, ie. by reducing the number of dimensions, without much loss of information. This technique used in image compression, as we will see in a later section.

This chapter will take you through the steps you needed to perform a Principal Components Analysis on a set of data. I am not going to describe exactly *why* the technique works, but I will try to provide an explanation of what is happening at each point so that you can make informed decisions when you try to use this technique yourself.

3.1 Method

Step 1: Get some data

In my simple example, I am going to use my own made-up data set. It's only got 2 dimensions, and the reason why I have chosen this is so that I can provide plots of the data to show what the PCA analysis is doing at each step.

The data I have used is found in Figure 3.1, along with a plot of that data.

Step 2: Subtract the mean

For PCA to work properly, you have to subtract the mean from each of the data dimensions. The mean subtracted is the average across each dimension. So, all the x values have $\bar{x}$ (the mean of the x values of all the data points) subtracted, and all the y values have $\bar{y}$ subtracted from them. This produces a data set whose mean is zero.

Data =

x	*y*
2.5	2.4
0.5	0.7
2.2	2.9
1.9	2.2
3.1	3.0
2.3	2.7
2	1.6
1	1.1
1.5	1.6
1.1	0.9

DataAdjust =

x	*y*
.69	.49
-1.31	-1.21
.39	.99
.09	.29
1.29	1.09
.49	.79
.19	-.31
-.81	-.81
-.31	-.31
-.71	-1.01

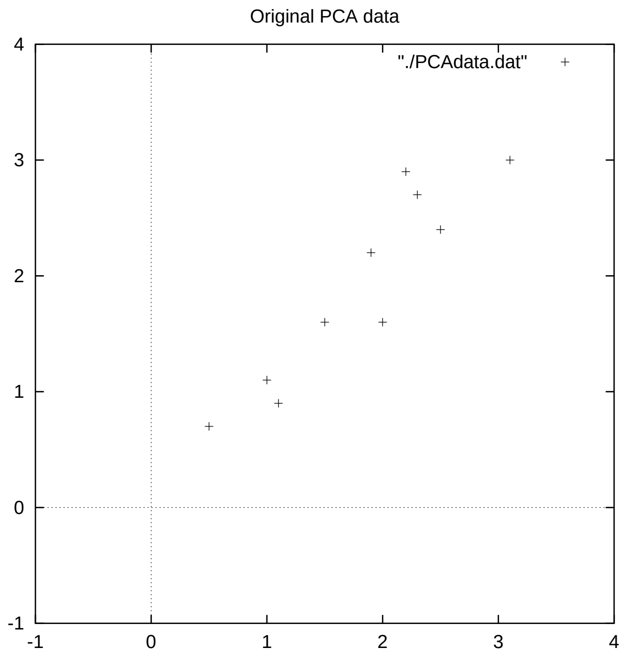


Figure 3.1: PCA example data, original data on the left, data with the means subtracted on the right, and a plot of the data

Step 3: Calculate the covariance matrix

This is done in exactly the same way as was discussed in section 2.1.4. Since the data is 2 dimensional, the covariance matrix will be 2×2. There are no surprises here, so I will just give you the result:

$$cov = \begin{pmatrix} .616555556 & .615444444 \\ .615444444 & .716555556 \end{pmatrix}$$

So, since the non-diagonal elements in this covariance matrix are positive, we should expect that both the x and y variable increase together.

Step 4: Calculate the eigenvectors and eigenvalues of the covariance matrix

Since the covariance matrix is square, we can calculate the eigenvectors and eigenvalues for this matrix. These are rather important, as they tell us useful information about our data. I will show you why soon. In the meantime, here are the eigenvectors and eigenvalues:

$$eigenvalues = \begin{pmatrix} .0490833989 \\ 1.28402771 \end{pmatrix}$$

$$eigenvectors = \begin{pmatrix} -.735178656 & -.677873399 \\ .677873399 & -.735178656 \end{pmatrix}$$

It is important to notice that these eigenvectors are both *unit* eigenvectors ie. their lengths are both 1. This is very important for PCA, but luckily, most maths packages, when asked for eigenvectors, will give you unit eigenvectors.

So what do they mean? If you look at the plot of the data in Figure 3.2 then you can see how the data has quite a strong pattern. As expected from the covariance matrix, they two variables do indeed increase together. On top of the data I have plotted both the eigenvectors as well. They appear as diagonal dotted lines on the plot. As stated in the eigenvector section, they are perpendicular to each other. But, more importantly, they provide us with information about the patterns in the data. See how one of the eigenvectors goes through the middle of the points, like drawing a line of best fit? That eigenvector is showing us how these two data sets are related along that line. The second eigenvector gives us the other, less important, pattern in the data, that all the points follow the main line, but are off to the side of the main line by some amount.

So, by this process of taking the eigenvectors of the covariance matrix, we have been able to extract lines that characterise the data. The rest of the steps involve transforming the data so that it is expressed in terms of them lines.

Step 5: Choosing components and forming a feature vector

Here is where the notion of data compression and reduced dimensionality comes into it. If you look at the eigenvectors and eigenvalues from the previous section, you

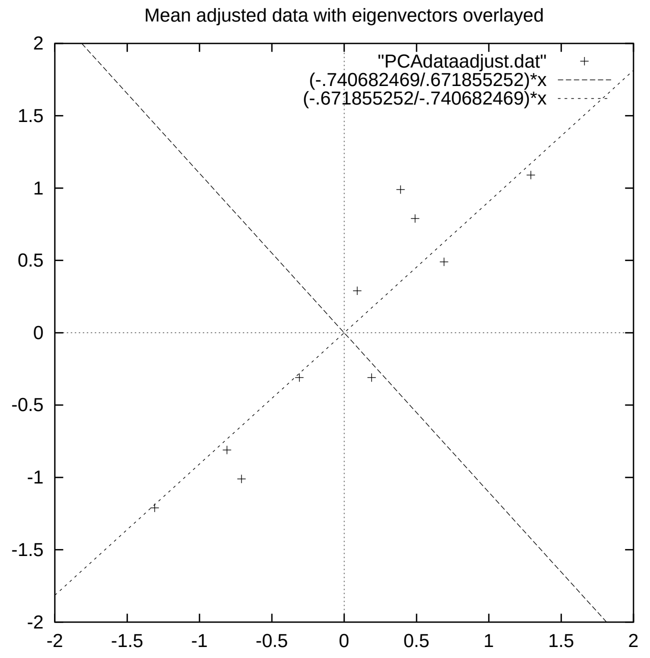


Figure 3.2: A plot of the normalised data (mean subtracted) with the eigenvectors of the covariance matrix overlayed on top.

will notice that the eigenvalues are quite different values. In fact, it turns out that the eigenvector with the *highest* eigenvalue is the *principle component* of the data set. In our example, the eigenvector with the larges eigenvalue was the one that pointed down the middle of the data. It is the most significant relationship between the data dimensions.

In general, once eigenvectors are found from the covariance matrix, the next step is to order them by eigenvalue, highest to lowest. This gives you the components in order of significance. Now, if you like, you can decide to *ignore* the components of lesser significance. You do lose some information, but if the eigenvalues are small, you don't lose much. If you leave out some components, the final data set will have less dimensions than the original. To be precise, if you originally have n dimensions in your data, and so you calculate n eigenvectors and eigenvalues, and then you choose only the first p eigenvectors, then the final data set has only p dimensions.

What needs to be done now is you need to form a *feature vector*, which is just a fancy name for a matrix of vectors. This is constructed by taking the eigenvectors that you want to keep from the list of eigenvectors, and forming a matrix with these eigenvectors in the columns.

$$FeatureVector = (eig_1\ eig_2\ eig_3\ ....\ eig_n)$$

Given our example set of data, and the fact that we have 2 eigenvectors, we have two choices. We can either form a feature vector with both of the eigenvectors:

$$\begin{pmatrix} -.677873399 & -.735178656 \\ -.735178656 & .677873399 \end{pmatrix}$$

or, we can choose to leave out the smaller, less significant component and only have a single column:

$$\begin{pmatrix} -.677873399 \\ -.735178656 \end{pmatrix}$$

We shall see the result of each of these in the next section.

Step 5: Deriving the new data set

This the final step in PCA, and is also the easiest. Once we have chosen the components (eigenvectors) that we wish to keep in our data and formed a feature vector, we simply take the transpose of the vector and multiply it on the left of the original data set, transposed.

$$FinalData = RowFeatureVector \times RowDataAdjust,$$

where $RowFeatureVector$ is the matrix with the eigenvectors in the columns *transposed* so that the eigenvectors are now in the rows, with the most significant eigenvector at the top, and $RowDataAdjust$ is the mean-adjusted data *transposed*, ie. the data items are in each column, with each row holding a separate dimension. I'm sorry if this sudden transpose of all our data confuses you, but the equations from here on are easier if we take the transpose of the feature vector and the data first, rather that having a little T symbol above their names from now on. $FinalData$ is the final data set, with data items in columns, and dimensions along rows.

What will this give us? It will give us the original data *solely in terms of the vectors we chose*. Our original data set had two axes, x and y, so our data was in terms of them. It is possible to express data in terms of any two axes that you like. If these axes are perpendicular, then the expression is the most efficient. This was why it was important that eigenvectors are always perpendicular to each other. We have changed our data from being in terms of the axes x and y, and now they are in terms of our 2 eigenvectors. In the case of when the new data set has reduced dimensionality, ie. we have left some of the eigenvectors out, the new data is only in terms of the vectors that we decided to keep.

To show this on our data, I have done the final transformation with each of the possible feature vectors. I have taken the transpose of the result in each case to bring the data back to the nice table-like format. I have also plotted the final points to show how they relate to the components.

In the case of keeping both eigenvectors for the transformation, we get the data and the plot found in Figure 3.3. This plot is basically the original data, rotated so that the eigenvectors are the axes. This is understandable since we have lost no information in this decomposition.

The other transformation we can make is by taking only the eigenvector with the largest eigenvalue. The table of data resulting from that is found in Figure 3.4. As expected, it only has a single dimension. If you compare this data set with the one resulting from using both eigenvectors, you will notice that this data set is exactly the first column of the other. So, if you were to plot this data, it would be 1 dimensional, and would be points on a line in exactly the x positions of the points in the plot in Figure 3.3. We have effectively thrown away the whole other axis, which is the other eigenvector.

So what have we done here? Basically we have transformed our data so that is expressed in terms of the patterns between them, where the patterns are the lines that most closely describe the relationships between the data. This is helpful because we have now classified our data point as a combination of the contributions from each of those lines. Initially we had the simple x and y axes. This is fine, but the x and y values of each data point don't really tell us exactly how that point relates to the rest of the data. Now, the values of the data points tell us exactly where (ie. above/below) the trend lines the data point sits. In the case of the transformation using *both* eigenvectors, we have simply altered the data so that it is in terms of those eigenvectors instead of the usual axes. But the single-eigenvector decomposition has removed the contribution due to the smaller eigenvector and left us with data that is only in terms of the other.

3.1.1 Getting the old data back

Wanting to get the original data back is obviously of great concern if you are using the PCA transform for data compression (an example of which to will see in the next section). This content is taken from
http://www.vision.auc.dk/ sig/Teaching/Flerdim/Current/hotelling/hotelling.html

Transformed Data=

x	y
-.827970186	-.175115307
1.77758033	.142857227
-.992197494	.384374989
-.274210416	.130417207
-1.67580142	-.209498461
-.912949103	.175282444
.0991094375	-.349824698
1.14457216	.0464172582
.438046137	.0177646297
1.22382056	-.162675287

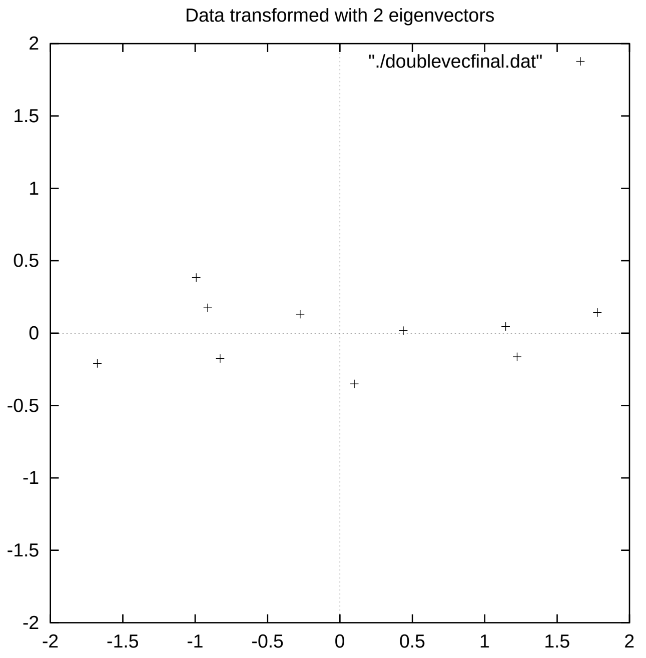


Figure 3.3: The table of data by applying the PCA analysis using both eigenvectors, and a plot of the new data points.

Transformed Data (Single eigenvector)

x
-.827970186
1.77758033
-.992197494
-.274210416
-1.67580142
-.912949103
.0991094375
1.14457216
.438046137
1.22382056

Figure 3.4: The data after transforming using only the most significant eigenvector

So, how do we get the original data back? Before we do that, remember that only if we took *all* the eigenvectors in our transformation will we get *exactly* the original data back. If we have reduced the number of eigenvectors in the final transformation, then the retrieved data has lost some information.

Recall that the final transform is this:

$$FinalData = RowFeatureVector \times RowDataAdjust,$$

which can be turned around so that, to get the original data back,

$$RowDataAdjust = RowFeatureVector^{-1} \times FinalData$$

where $RowFeatureVector^{-1}$ is the inverse of $RowFeatureVector$. However, when we take *all* the eigenvectors in our feature vector, it turns out that the inverse of our feature vector is actually equal to the transpose of our feature vector. This is only true because the elements of the matrix are all the unit eigenvectors of our data set. This makes the return trip to our data easier, because the equation becomes

$$RowDataAdjust = RowFeatureVector^{T} \times FinalData$$

But, to get the actual original data back, we need to add on the mean of that original data (remember we subtracted it right at the start). So, for completeness,

$$RowOriginalData = (RowFeatureVector^{T} \times FinalData) + OriginalMean$$

This formula also applies to when you do not have all the eigenvectors in the feature vector. So even when you leave out some eigenvectors, the above equation still makes the correct transform.

I will not perform the data re-creation using the *complete* feature vector, because the result is exactly the data we started with. However, I will do it with the reduced feature vector to show you how information has been lost. Figure 3.5 show this plot. Compare

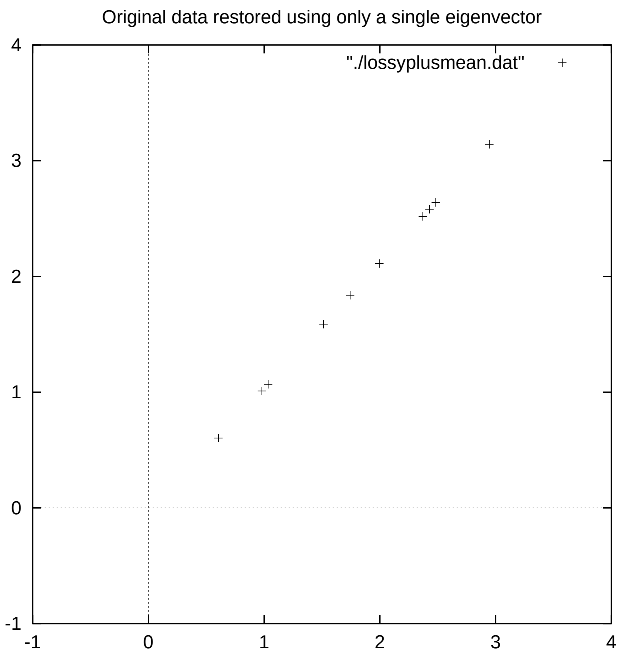


Figure 3.5: The reconstruction from the data that was derived using only a single eigenvector

it to the original data plot in Figure 3.1 and you will notice how, while the variation along the principle eigenvector (see Figure 3.2 for the eigenvector overlayed on top of the mean-adjusted data) has been kept, the variation along the other component (the other eigenvector that we left out) has gone.

Exercises

- What do the eigenvectors of the covariance matrix give us?
- At what point in the PCA process can we decide to compress the data? What effect does this have?
- For an example of PCA and a graphical representation of the principal eigenvectors, research the topic 'Eigenfaces', which uses PCA to do facial recognition

Chapter 4

Application to Computer Vision

This chapter will outline the way that PCA is used in computer vision, first showing how images are usually represented, and then showing what PCA can allow us to do with those images. The information in this section regarding facial recognition comes from "Face Recognition: Eigenface, Elastic Matching, and Neural Nets", Jun Zhang et al. Proceedings of the IEEE, Vol. 85, No. 9, September 1997. The representation information, is taken from "Digital Image Processing" Rafael C. Gonzalez and Paul Wintz, Addison-Wesley Publishing Company, 1987. It is also an excellent reference for further information on the K-L transform in general. The image compression information is taken from *http://www.vision.auc.dk/ sig/Teaching/Flerdim/Current/hotelling/hotelling.html*, which also provides examples of image reconstruction using a varying amount of eigenvectors.

4.1 Representation

When using these sort of matrix techniques in computer vision, we must consider representation of images. A square, N by N image can be expressed as an N^2-dimensional vector

$$X = \left(\begin{array}{cccccc} x_1 & x_2 & x_3 & . & . & x_{N^2} \end{array} \right)$$

where the rows of pixels in the image are placed one after the other to form a one-dimensional image. E.g. The first N elements ($x_1 - x_N$ will be the first row of the image, the next N elements are the next row, and so on. The values in the vector are the intensity values of the image, possibly a single greyscale value.

4.2 PCA to find patterns

Say we have 20 images. Each image is N pixels high by N pixels wide. For each image we can create an image vector as described in the representation section. We can then put all the images together in one big image-matrix like this:

$$ImagesMatrix = \begin{pmatrix} ImageVec1 \\ ImageVec2 \\ . \\ . \\ ImageVec20 \end{pmatrix}$$

which gives us a starting point for our PCA analysis. Once we have performed PCA, we have our original data in terms of the eigenvectors we found from the covariance matrix. Why is this useful? Say we want to do facial recognition, and so our original images were of peoples faces. Then, the problem is, given a new image, whose face from the original set is it? (Note that the new image is not one of the 20 we started with.) The way this is done is computer vision is to measure the difference between the new image and the original images, but not along the original axes, along the new axes derived from the PCA analysis.

It turns out that these axes works much better for recognising faces, because the PCA analysis has given us the original images *in terms of the differences and similarities between them*. The PCA analysis has identified the statistical patterns in the data.

Since all the vectors are N^2 dimensional, we will get N^2 eigenvectors. In practice, we are able to leave out some of the less significant eigenvectors, and the recognition still performs well.

4.3 PCA for image compression

Using PCA for image compression also know as the Hotelling, or Karhunen and Leove (KL), transform. If we have 20 images, each with N^2 pixels, we can form N^2 vectors, each with 20 dimensions. Each vector consists of all the intensity values from the *same* pixel from each picture. This is different from the previous example because before we had a vector for *image*, and each item in that vector was a different pixel, whereas now we have a vector for each *pixel*, and each item in the vector is from a different image.

Now we perform the PCA on this set of data. We will get 20 eigenvectors because each vector is 20-dimensional. To compress the data, we can then choose to transform the data only using, say 15 of the eigenvectors. This gives us a final data set with only 15 dimensions, which has saved us $1/4$ of the space. However, when the original data is reproduced, the images have lost some of the information. This compression technique is said to be *lossy* because the decompressed image is not exactly the same as the original, generally worse.

Appendix A

Implementation Code

This is code for use in Scilab, a freeware alternative to Matlab. I used this code to generate all the examples in the text. Apart from the first macro, all the rest were written by me.

```
// This macro taken from
// http://www.cs.montana.edu/˜harkin/courses/cs530/scilab/macros/cov.sci
// No alterations made

// Return the covariance matrix of the data in x, where each column of x
// is one dimension of an n-dimensional data set.  That is, x has x columns
// and m rows, and each row is one sample.
//
// For example, if x is three dimensional and there are 4 samples.
//   x = [1 2 3;4 5 6;7 8 9;10 11 12]
//   c = cov (x)

function [c]=cov (x)
// Get the size of the array
sizex=size(x);
// Get the mean of each column
meanx = mean (x, "r");
// For each pair of variables, x1, x2, calculate
// sum ((x1 - meanx1)(x2-meanx2))/(m-1)
for var = 1:sizex(2),
        x1 = x(:,var);
        mx1 = meanx (var);
        for ct = var:sizex (2),
                x2 = x(:,ct);
                mx2 = meanx (ct);
                v = ((x1 - mx1)’ * (x2 - mx2))/(sizex(1) - 1);
                cv(var,ct) = v;
                cv(ct,var) = v;
                // do the lower part of c also.
        end,
end,
c=cv;


// This a simple wrapper function to get just the eigenvectors
// since the system call returns 3 matrices
function [x]=justeigs (x)
// This just returns the eigenvectors of the matrix

[a, eig, b] = bdiag(x);

x= eig;



// this function makes the transformation to the eigenspace for PCA
// parameters:
// adjusteddata = mean-adjusted data set
// eigenvectors = SORTED eigenvectors (by eigenvalue)
// dimensions  = how many eigenvectors you wish to keep
//
// The first two parameters can come from the result of calling
// PCAprepare on your data.
// The last is up to you.

function [finaldata] = PCAtransform(adjusteddata,eigenvectors,dimensions)
finaleigs = eigenvectors(:,1:dimensions);
prefinaldata = finaleigs'*adjusteddata';
finaldata = prefinaldata';



// This function does the preparation for PCA analysis
// It adjusts the data to subtract the mean, finds the covariance matrix,
// and finds normal eigenvectors of that covariance matrix.
// It returns 4 matrices
// meanadjust = the mean-adjust data set
// covmat = the covariance matrix of the data
// eigvalues = the eigenvalues of the covariance matrix, IN SORTED ORDER
// normaleigs = the normalised eigenvectors of the covariance matrix,
// IN SORTED ORDER WITH RESPECT TO
// THEIR EIGENVALUES, for selection for the feature vector.
```

```
//
// NOTE: This function cannot handle data sets that have any eigenvalues
// equal to zero.  It’s got something to do with the way that scilab treats
// the empty matrix and zeros.
//
function [meanadjusted,covmat,sorteigvalues,sortnormaleigs] = PCAprepare (data)
// Calculates the mean adjusted matrix, only for 2 dimensional data
means = mean(data,"r");
meanadjusted = meanadjust(data);
covmat = cov(meanadjusted);
eigvalues = spec(covmat);
normaleigs = justeigs(covmat);
sorteigvalues = sorteigvectors(eigvalues’,eigvalues’);
sortnormaleigs = sorteigvectors(eigvalues’,normaleigs);



// This removes a specified column from a matrix
// A = the matrix
// n = the column number you wish to remove
function [columnremoved] = removecolumn(A,n)
inputsize = size(A);
numcols = inputsize(2);
temp = A(:,1:(n-1));
for var = 1:(numcols - n)
        temp(:,(n+var)-1) = A(:,(n+var));
end,
columnremoved = temp;


// This finds the column number that has the
// highest value in it’s first row.
function [column] = highestvalcolumn(A)
inputsize = size(A);
numcols = inputsize(2);
maxval = A(1,1);
maxcol = 1;
for var = 2:numcols
        if A(1,var) > maxval
                maxval = A(1,var);
                maxcol = var;
        end,
end,
column = maxcol
```

```
// This sorts a matrix of vectors, based on the values of
// another matrix
//
// values = the list of eigenvalues (1 per column)
// vectors = The list of eigenvectors (1 per column)
//
// NOTE:  The values should correspond to the vectors
// so that the value in column x corresponds to the vector
// in column x.
function [sortedvecs] = sorteigvectors(values,vectors)
inputsize = size(values);
numcols  = inputsize(2);
highcol = highestvalcolumn(values);
sorted = vectors(:,highcol);
remainvec = removecolumn(vectors,highcol);
remainval = removecolumn(values,highcol);
for var = 2:numcols
        highcol = highestvalcolumn(remainval);
        sorted(:,var) = remainvec(:,highcol);
        remainvec = removecolumn(remainvec,highcol);
        remainval = removecolumn(remainval,highcol);
end,
sortedvecs = sorted;


// This takes a set of data, and subtracts
// the column mean from each column.
function [meanadjusted] = meanadjust(Data)
inputsize = size(Data);
numcols = inputsize(2);
means = mean(Data,"r");
tmpmeanadjusted = Data(:,1) - means(:,1);
for var = 2:numcols
        tmpmeanadjusted(:,var) = Data(:,var) - means(:,var);
end,
meanadjusted = tmpmeanadjusted
```

Singular Value Decomposition Tutorial

Kirk Baker

March 29, 2005 (Revised January 14, 2013)

Contents

1 Acknowledgments

Several people have been kind enough to point out errors in the original document or otherwise provide encouragement, including Xiaofei Lu, John McNally, Maiko Sell, Li Xue, don quitoxe, Juda Djatmiko, Dio Coumans, Mansi Radke, Roberto Mirizzi, Gursimran singh, Ryan Angilly, Mark Wong-VanHaren, Harry Podschwit, Laxmi, Paul Hurt, Pritesh Patel, and B. L. Deekshatulu. I tried to fix all the typos, but haven't gotten around to the more substantive suggestions yet. Thank you all.

I wrote this as an assignment for an NLP seminar taught by Chris Brew. Thank you can never suffice.

2 Introduction

Most tutorials on complex topics are apparently written by very smart people whose goal is to use as little space as possible and who assume that their readers already know almost as much as the author does. This tutorial's not like that. It's more a manifestivus for the rest of us. It's about the mechanics of singular value decomposition, especially as it relates to some techniques in natural language processing. It's written by someone who knew zilch about singular value decomposition or any of the underlying math before he started writing it, and knows barely more than that now. Accordingly, it's a bit long on the background part, and a bit short on the truly explanatory part, but hopefully it contains all the information necessary for someone who's never heard of singular value decomposition before to be able to do it.

3 Points and Space

A point is just a list of numbers. This list of numbers, or *coordinates*, specifies the point's position in space. How many coordinates there are determines the *dimensions* of that space.

For example, we can specify the position of a point on the edge of a ruler with a single coordinate. The position of the two points $0.5cm$ and $1.2cm$ are precisely specified by single coordinates. Because we're using a single coordinate to identify a point, we're dealing with points in one-dimensional space, or 1-space.

The position of a point anywhere in a plane is specified with a pair of coordinates; it takes three coordinates to locate points in three dimensions. Nothing stops us from going beyond points in 3-space. The fourth dimension is often used to indicate time, but the dimensions can be chosen to represent whatever measurement unit is relevant to the objects we're trying to describe.

Generally, space represented by more than three dimensions is called *hyperspace*. You'll also see the term *n-space* used to talk about spaces of different dimensionality (e.g. 1-space, 2-space, ..., n-space).

For example, if I want a succinct way of describing the amount of food I eat in a given day, I can use points in n-space to do so. Let the dimensions of this space be the following food items:

Eggs Grapes Bananas Chickens Cans of Tuna

There are five categories, so we're dealing with points in 5-space. Thus, the interpretation of the point $(3, 18, 2, 0.5, 1,)$ would be "three eggs, eighteen grapes, two bananas, half a chicken, one can of tuna".

4 Vectors

For most purposes, points and vectors are essentially the same thing[1], that is, a sequence of numbers corresponding to measurements along various dimensions.

Vectors are usually denoted by a lower case letter with an arrow on top, e.g. $\vec{x}$. The numbers comprising the vector are now called *components*, and the number of components equals the dimensionality of the vector. We use a subscript on the vector name to refer to the component in that position. In the example below, $\vec{x}$ is a 5-dimensional vector, $x_1 = 8$, $x_2 6$, etc.

$$\vec{x} = \begin{pmatrix} 8 \\ 6 \\ 7 \\ 5 \\ 3 \end{pmatrix}$$

Vectors can be equivalently represented horizontally to save space, e.g. $\vec{x} = [8, 6, 7, 5, 3]$ is the same vector as above. More generally, a vector $\vec{x}$ with n-dimensions is a sequence of n numbers, and component x_i represents the value of $\vec{x}$ on the i^{th} dimension.

[1]Technically, I think, a vector is a function that takes a point as input and returns as its value a point of the same dimensionality.

5 Matrices

A matrix is probably most familiar as a table of data, like Table 1, which shows the top 5 scorers on a judge's scorecard in the 1997 Fitness International competition.

Contestant	Round 1	Round 2	Round 3	Round 4	Total	Place
Carol Semple-Marzetta	17	18	5	5	45	1
Susan Curry	42	28	30	15	115	3
Monica Brant	10	10	10	21	51	2
Karen Hulse	28	5	65	39	132	5
Dale Tomita	24	26	45	21	116	4

Table 1: 1997 Fitness International Scorecard. Source: Muscle & Fitness July 1997, p.139

A table consists of rows (the horizontal list of scores corresponding to a contestant's name), and columns (the vertical list of numbers corresponding to the scores for a given round). What makes this table a matrix is that it's a rectangular array of numbers. Written as a matrix, Table 1 looks like this:

$$\begin{bmatrix} 17 & 18 & 5 & 5 & 45 & 1 \\ 42 & 28 & 30 & 15 & 115 & 3 \\ 10 & 10 & 10 & 21 & 51 & 2 \\ 28 & 5 & 65 & 39 & 132 & 5 \\ 24 & 26 & 45 & 21 & 116 & 4 \end{bmatrix}$$

The size, or dimensions, of a matrix is given in terms of the number of rows by the number of columns. This makes the matrix above a *"five by six"* matrix, written 5×6 matrix.

We can generalize the descriptions made so far by using variables to stand in for the actual numbers we've been using. Traditionally, a matrix in the abstract is named A. The maximum number of rows is assigned to the variable m, and the number of columns is called n. Matrix *entries* (also called *elements* or *components*) are denoted by a lower-case a, and a particular entry is referenced by its row index (labeled i) and its column index (labeled j). For example, 132 is the entry in row 4 and column 5 in the matrix above, so another way of saying that would be $a_{45} = 132$. More generally, the element in the i_{th} row and j_{th} column is labeled a_{ij}, and called the ij-entry or ij-component.

A little more formally than before, we can denote a matrix like this:

5.1 Matrix Notation

Let m, n be two integers ≥ 1. Let a_{ij}, $i = 1, ..., m$, $j = 1, ..., n$ be mn numbers. An array of numbers

$$A = \begin{bmatrix} a_{11} & ... & a_{1j} & ... & a_{1n} \\ . & & . & & . \\ a_{i1} & ... & a_{ij} & ... & a_{in} \\ . & & . & & . \\ a_{m1} & ... & a_{mj} & ... & a_{mn} \end{bmatrix}$$

is an $m \times n$ matrix and the numbers a_{ij} are elements of A. The sequence of numbers

$$A_{(i)} = (a_{i1}, ..., a_{in})$$

is the i_{th} row of A, and the sequence of numbers

$$A^{(j)} = (a_{1j}, ..., a_{mj})$$

is the j_{th} column of A.

Just as the distinction between points and vectors can blur in practice, so does the distinction between vectors and matrices. A matrix is basically a collection of vectors. We can talk about row vectors or column vectors. Or a vector with n components can be considered a $1 \times n$ matrix.

For example, the matrix below is a word×document matrix which shows the number of times a particular word occurs in some made-up documents. Typical accompanying descrip-

	Doc 1	Doc 2	Doc 3
abbey	2	3	5
spinning	1	0	1
soil	3	4	1
stunned	2	1	3
wrath	1	1	4

Table 2: Word×document matrix for some made-up documents.

tions of this kind of matrix might be something like “high dimensional vector space model”. The dimensions are the words, if we’re talking about the column vectors representing documents, or documents, if we’re talking about the row vectors which represent words. High dimensional means we have a lot of them. Thus, “hyperspace document representation” means a document is represented as a vector whose components correspond in some way to the words in it, plus there are a lot of words. This is equivalent to “a document is represented as a point in n-dimensional space.”

6 Vector Terminology

6.1 Vector Length

The length of a vector is found by squaring each component, adding them all together, and taking the square root of the sum. If $\vec{v}$ is a vector, its length is denoted by $|\vec{v}|$. More concisely,

$$|\vec{v}| = \sqrt{\sum_{i=1}^{n} v_i^2}$$

For example, if $\vec{v} = [4, 11, 8, 10]$, then

$$|\vec{v}| = \sqrt{4^2 + 11^2 + 8^2 + 10^2} = \sqrt{301} = 17.35$$

6.2 Vector Addition

Adding two vectors means adding each component in $\vec{v_1}$ to the component in the corresponding position in $\vec{v_2}$ to get a new vector. For example

$$[3, 2, 1, -2] + [2, -1, 4, 1] = [(3 + 2), (2 - 1), (1 + 4), (-2 + 1)] = [5, 1, 5, -1]$$

More generally, if $A = [a_1, a_2, ...a_n]$ and $B = [b_1, b_2, ...b_n]$, then $A+B = [a_1+b_1, a_2+b_2, ...a_n+b_n]$.

6.3 Scalar Multiplication

Multiplying a scalar (real number) times a vector means multiplying every component by that real number to yield a new vector. For instance, if $\vec{v} = [3, 6, 8, 4]$, then $1.5 * \vec{v} = 1.5 * [3, 6, 8, 4] = [4.5, 9, 12, 6]$. More generally, *scalar multiplication* means if d is a real number and $\vec{v}$ is a vector $[v_1, v_2, ..., v_n]$, then $d * \vec{v} = [dv_1, dv_2, ..., dv_n]$.

6.4 Inner Product

The *inner product* of two vectors (also called the *dot product* or *scalar product*) defines multiplication of vectors. It is found by multiplying each component in $\vec{v_1}$ by the component in $\vec{v_2}$ in the same position and adding them all together to yield a scalar value. The inner product is only defined for vectors of the same dimension. The inner product of two vectors is denoted $(\vec{v_1}, \vec{v_2})$ or $\vec{v_1} \cdot \vec{v_2}$ (the dot product). Thus,

$$(\vec{x}, \vec{y}) = \vec{x} \cdot \vec{y} = \sum_{i=1}^{n} x_i y_i$$

For example, if $\vec{x} = [1, 6, 7, 4]$ and $\vec{y} = [3, 2, 8, 3]$, then

$$\vec{x} \cdot \vec{y} = 1(3) + 6(2) + 7(8) + 3(4) = 83$$

6.5 Orthogonality

Two vectors are *orthogonal* to each other if their inner product equals zero. In two-dimensional space this is equivalent to saying that the vectors are perpendicular, or that the only angle between them is a 90° angle. For example, the vectors $[2, 1, -2, 4]$ and $[3, -6, 4, 2]$ are orthogonal because

$$[2, 1, -2, 4] \cdot [3, -6, 4, 2] = 2(3) + 1(-6) - 2(4) + 4(2) = 0$$

6.6 Normal Vector

A *normal vector* (or *unit vector*) is a vector of length 1. Any vector with an initial length $>$ 0 can be normalized by dividing each component in it by the vector's length. For example, if $\vec{v} = [2, 4, 1, 2]$, then

$$|\vec{v}| = \sqrt{2^2 + 4^2 + 1^2 + 2^2} = \sqrt{25} = 5$$

Then $\vec{u} = [2/5, 4/5, 1/5, 1/5]$ is a normal vector because

$$|\vec{u}| = \sqrt{(2/5)^2 + (4/5)^2 + (1/5)^2 + (2/5)^2} = \sqrt{25/25} = 1$$

6.7 Orthonormal Vectors

Vectors of unit length that are orthogonal to each other are said to be *orthonormal.* For example,

$$\vec{u} = [2/5, 1/5, -2/5, 4/5]$$

and

$$\vec{v} = [3/\sqrt{65}, -6/\sqrt{65}, 4/\sqrt{65}, 2/\sqrt{65}]$$

are orthonormal because

$$|\vec{u}| = \sqrt{(2/5)^2 + (1/5)^2 + (-2/5)^2 + (4/5)^2} = 1$$

$$|\vec{v}| = \sqrt{(3/\sqrt{65})^2 + (-6/\sqrt{65})^2 + (4/\sqrt{65})^2 + (2/\sqrt{65})^2} = 1$$

$$\vec{u} \cdot \vec{v} = \frac{6}{5\sqrt{65}} - \frac{6}{5\sqrt{65}} - \frac{8}{5\sqrt{65}} + \frac{8}{5\sqrt{65}} = 0$$

6.8 Gram-Schmidt Orthonormalization Process

The Gram-Schmidt orthonormalization process is a method for converting a set of vectors into a set of orthonormal vectors. It basically begins by normalizing the first vector under consideration and iteratively rewriting the remaining vectors in terms of themselves minus a multiplication of the already normalized vectors. For example, to convert the column vectors of

$$A = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 2 & 0 \\ 2 & 3 & 1 \\ 1 & 1 & 0 \end{bmatrix}$$

into orthonormal column vectors

$$A = \begin{bmatrix} \frac{\sqrt{6}}{6} & \frac{\sqrt{2}}{6} & \frac{2}{3} \\ 0 & \frac{2\sqrt{2}}{3} & \frac{-1}{3} \\ \frac{\sqrt{6}}{3} & 0 & 0 \\ \frac{\sqrt{6}}{6} & \frac{-\sqrt{2}}{6} & \frac{-2}{3} \end{bmatrix},$$

first normalize $\vec{v_1} = [1, 0, 2, 1]$:

$$\vec{u_1} = [\frac{1}{\sqrt{6}}, 0, \frac{2}{\sqrt{6}}, \frac{1}{\sqrt{6}}].$$

Next, let

$$\vec{w_2} = \vec{v_2} - \vec{u_1} \cdot \vec{v_2} * \vec{u_1} = [2,2,3,1] - [\frac{1}{\sqrt{6}}, 0, \frac{2}{\sqrt{6}}, \frac{1}{\sqrt{6}}] \cdot [2,2,3,1] * [\frac{1}{\sqrt{6}}, 0, \frac{2}{\sqrt{6}}, \frac{1}{\sqrt{6}}]$$

$$= [2,2,3,1] - (\frac{9}{\sqrt{6}}) * [\frac{1}{\sqrt{6}}, 0, \frac{2}{\sqrt{6}}, \frac{1}{\sqrt{6}}]$$

$$= [2,2,3,1] - [\frac{3}{2}, 0, 3, \frac{3}{2}]$$

$$= [\frac{1}{2}, 2, 0, \frac{-1}{2}]$$

Normalize $\vec{w_2}$ to get

$$\vec{u_2} = [\frac{\sqrt{2}}{6}, \frac{2\sqrt{2}}{3}, 0, \frac{-\sqrt{2}}{6}]$$

Now compute $\vec{u_3}$ in terms of $\vec{u_1}$ and $\vec{u_2}$ as follows. Let

$$\vec{w_3} = \vec{v_3} - \vec{u_1} \cdot \vec{v_3} * \vec{u_1} - \vec{u_2} \cdot \vec{v_3} * \vec{u_2} = [\frac{4}{9}, \frac{-2}{9}, 0, \frac{-4}{9}]$$

and normalize $\vec{w_3}$ to get

$$\vec{u_3} = [\frac{2}{3}, \frac{-1}{3}, 0, \frac{-2}{3}]$$

More generally, if we have an orthonormal set of vectors $\vec{u_1}, .., \vec{u_{k-1}}$, then $\vec{w_k}$ is expressed as

$$\vec{w_k} = \vec{v_k} - \sum_{i=1}^{k-1} \vec{u_i} \cdot \vec{v_k} * \vec{u_i}$$

7 Matrix Terminology

7.1 Square Matrix

A matrix is said to be *square* if it has the same number of rows as columns. To designate the size of a square matrix with n rows and columns, it is called *n-square.* For example, the matrix below is 3-square.

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

7.2 Transpose

The *transpose* of a matrix is created by converting its rows into columns; that is, row 1 becomes column 1, row 2 becomes column 2, etc. The transpose of a matrix is indicated with a superscripted ^{T}, e.g. the transpose of matrix A is A^T. For example, if

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$$

then its transpose is

$$A^T = \begin{bmatrix} 1 & 4 \\ 2 & 5 \\ 3 & 6 \end{bmatrix}$$

7.3 Matrix Multiplication

It is possible to multiply two matrices only when the second matrix has the same number of rows as the first matrix has columns. The resulting matrix has as many rows as the first matrix and as many columns as the second matrix. In other words, if A is a $m \times n$ matrix and B is a $n \times s$ matrix, then the product AB is an $m \times s$ matrix.

The coordinates of AB are determined by taking the inner product of each row of A and each column in B. That is, if $A_1, ..., A_m$ are the row vectors of matrix A, and $B^1, ..., B^s$ are the column vectors of B, then ab_{ik} of AB equals $A_i \cdot B^k$. The example below illustrates.

$$A = \begin{bmatrix} 2 & 1 & 4 \\ 1 & 5 & 2 \end{bmatrix} B = \begin{bmatrix} 3 & 2 \\ -1 & 4 \\ 1 & 2 \end{bmatrix} AB = \begin{bmatrix} 2 & 1 & 4 \\ 1 & 5 & 2 \end{bmatrix} \begin{bmatrix} 3 & 2 \\ -1 & 4 \\ 1 & 2 \end{bmatrix} = \begin{bmatrix} 9 & 16 \\ 0 & 26 \end{bmatrix}$$

$$ab_{11} = \begin{bmatrix} 2 & 1 & 4 \end{bmatrix} \begin{bmatrix} 3 \\ -1 \\ 1 \end{bmatrix} = 2(3) + 1(-1) + 4(1) = 9$$

$$ab_{12} = \begin{bmatrix} 2 & 1 & 4 \end{bmatrix} \begin{bmatrix} 2 \\ 4 \\ 2 \end{bmatrix} = 2(4) + 1(4) + 4(2) = 16$$

$$ab_{21} = \begin{bmatrix} 1 & 5 & 2 \end{bmatrix} \begin{bmatrix} 3 \\ -1 \\ 1 \end{bmatrix} = 1(3) + 5(-1) + 2(1) = 0$$

$$ab_{22} = \begin{bmatrix} 1 & 5 & 2 \end{bmatrix} \begin{bmatrix} 2 \\ 4 \\ 2 \end{bmatrix} = 1(2) + 5(4) + 2(2) = 26$$

7.4 Identity Matrix

The identity matrix is a square matrix with entries on the diagonal equal to 1 and all other entries equal zero. The diagonal is all the entries a_{ij} where $_i =_j$, i.e., $a_{11}, a_{22}, ..., a_{mm}$. The n-square identity matrix is denoted variously as $I_{n \times n}$, I_n, or simply I. The identity matrix behaves like the number 1 in ordinary multiplication, which mean $AI = A$, as the example below shows.

$$A = \begin{bmatrix} 2 & 4 & 6 \\ 1 & 3 & 5 \end{bmatrix} I = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} AI = \begin{bmatrix} 2 & 4 & 6 \\ 1 & 3 & 5 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} =$$

$$ai_{11} = \begin{bmatrix} 2 & 4 & 6 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} = 2(1) + 0(4) + 0(6) = 2$$

$$ai_{12} = \begin{bmatrix} 2 & 4 & 6 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} = 2(0) + 4(1) + 6(0) = 4$$

$$ai_{13} = \begin{bmatrix} 2 & 4 & 6 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} = 2(0) + 4(0) + 6(1) = 6$$

$$ai_{21} = \begin{bmatrix} 1 & 3 & 5 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} = 1(1) + 3(0) + 5(0) = 1$$

$$ai_{22} = \begin{bmatrix} 1 & 3 & 5 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} = 1(0) + 3(1) + 5(0) = 3$$

$$ai_{23} = \begin{bmatrix} 1 & 3 & 5 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} = 1(0) + 3(0) + 5(1) = 5$$

$$= \begin{bmatrix} 2 & 4 & 6 \\ 1 & 3 & 5 \end{bmatrix}$$

7.5 Orthogonal Matrix

A matrix A is orthogonal if $AA^T = A^T A = I$. For example,

$$A = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 3/5 & -4/5 \\ 0 & 4/5 & 3/5 \end{bmatrix}$$

is orthogonal because

$$A^T A = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 3/5 & -4/5 \\ 0 & 4/5 & 3/5 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 3/5 & 4/5 \\ 0 & -4/5 & 3/5 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

7.6 Diagonal Matrix

A diagonal matrix A is a matrix where all the entries ai_{ij} are 0 when $i \neq j$. In other words, the only nonzero values run along the main dialog from the upper left corner to the lower right corner:

$$A = \begin{bmatrix} a_{11} & 0 & ... & 0 \\ 0 & a_{22} & & 0 \\ . & & . & . \\ 0 & ... & & a_{mm} \end{bmatrix}$$

7.7 Determinant

A determinant is a function of a square matrix that reduces it to a single number. The determinant of a matrix A is denoted $|A|$ or $det(A)$. If A consists of one element a, then $|A| = a$; in other words if $A = [6]$ then $|A| = 6$. If A is a 2×2 matrix, then

$$|A| = \begin{vmatrix} a & b \\ c & d \end{vmatrix} = ad - bc.$$

For example, the determinant of

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

is

$$|A| = \begin{vmatrix} 4 & 1 \\ 1 & 2 \end{vmatrix} = 4(2) - 1(1) = 7.$$

Finding the determinant of an n-square matrix for $n > 2$ can be done by recursively deleting rows and columns to create successively smaller matrices until they are all 2×2 dimensions, and then applying the previous definition. There are several tricks for doing this efficiently, but the most basic technique is called *expansion by row* and is illustrated below for a 3×3 matrix. In this case we are expanding by row 1, which means deleting row 1 and successively deleting columns 1, column 2, and column 3 to create three 2×2 matrices. The determinant of each smaller matrix is multiplied by the entry corresponding to the intersection of the deleted row and column. The expansion alternately adds and subtracts each successive determinant.

$$\begin{vmatrix} -1 & 4 & 3 \\ 2 & 6 & 4 \\ 3 & -2 & 8 \end{vmatrix} = (-1)\begin{vmatrix} 6 & 4 \\ -2 & 8 \end{vmatrix} - (4)\begin{vmatrix} 2 & 4 \\ 3 & 8 \end{vmatrix} + (3)\begin{vmatrix} 2 & 6 \\ 3 & -2 \end{vmatrix} =$$

$$-1(6 \cdot 8 - 4 \cdot -2) - 4(2 \cdot 8 - 4 \cdot 3) + 3(2 \cdot -2 - 3 \cdot 6) =$$

$$-56 - 16 - 66 = -138$$

The determinant of a 4×4 matrix would be found by expanding across row 1 to alternately add and subtract 4 3×3 determinants, which would themselves be expanded to produce a series of 2×2 determinants that would be reduced as above. This procedure can be applied to find the determinant of an arbitrarily large square matrix.

7.8 Eigenvectors and Eigenvalues

An *eigenvector* is a nonzero vector that satisfies the equation

$$A\vec{v} = \lambda\vec{v}$$

where A is a square matrix, λ is a scalar, and $\vec{v}$ is the eigenvector. λ is called an *eigenvalue*. Eigenvalues and eigenvectors are also known as, respectively, *characteristic roots* and *characteristic vectors*, or *latent roots* and *latent vectors*.

You can find eigenvalues and eigenvectors by treating a matrix as a system of linear equations and solving for the values of the variables that make up the components of the eigenvector. For example, finding the eigenvalues and corresponding eigenvectors of the matrix

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

means applying the above formula to get

$$A\vec{v} = \lambda\vec{v} = \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \lambda \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

in order to solve for λ, x_1 and x_2. This statement is equivalent to the system of equations

$$2x_1 + x_2 = \lambda x_1$$

$$x_1 + 2x_2 = \lambda x_2$$

which can be rearranged as

$$(2-\lambda)x_1 + x_2 = 0$$

$$x_1 + (2-\lambda)x_2 = 0$$

A necessary and sufficient condition for this system to have a nonzero vector $[x_1, x_2]$ is that the determinant of the coefficient matrix

$$\begin{bmatrix} (2-\lambda) & 1 \\ 1 & (2-\lambda) \end{bmatrix}$$

be equal to zero. Accordingly,

$$\begin{vmatrix} (2-\lambda) & 1 \\ 1 & (2-\lambda) \end{vmatrix} = 0$$

$$(2-\lambda)(2-\lambda) - 1 \cdot 1 = 0$$

$$\lambda^2 - 4\lambda + 3 = 0$$

$$(\lambda - 3)(\lambda - 1) = 0$$

There are two values of λ that satisfy the last equation; thus there are two eigenvalues of the original matrix A and these are $\lambda_1 = 3, \lambda_2 = 1$.

We can find eigenvectors which correspond to these eigenvalues by plugging λ back in to the equations above and solving for x_1 and x_2. To find an eigenvector corresponding to $\lambda = 3$, start with

$$(2-\lambda)x_1 + x_2 = 0$$

and substitute to get

$$(2-3)x_1 + x_2 = 0$$

which reduces and rearranges to

$$x_1 = x_2$$

There are an infinite number of values for x_1 which satisfy this equation; the only restriction is that not all the components in an eigenvector can equal zero. So if $x_1 = 1$, then $x_2 = 1$ and an eigenvector corresponding to $\lambda = 3$ is $[1, 1]$.

Finding an eigenvector for $\lambda = 1$ works the same way.

$$(2-1)x_1 + x_2 = 0$$

$$x_1 = -x_2$$

So an eigenvector for $\lambda = 1$ is $[1, -1]$.

8 Singular Value Decomposition

Singular value decomposition (SVD) can be looked at from three mutually compatible points of view. On the one hand, we can see it as a method for transforming correlated variables into a set of uncorrelated ones that better expose the various relationships among the original data items. At the same time, SVD is a method for identifying and ordering the dimensions along which data points exhibit the most variation. This ties in to the third way of viewing SVD, which is that once we have identified where the most variation is, it's possible to find the best approximation of the original data points using fewer dimensions. Hence, SVD can be seen as a method for data reduction.

As an illustration of these ideas, consider the 2-dimensional data points in Figure 1. The regression line running through them shows the best approximation of the original data with a 1-dimensional object (a line). It is the best approximation in the sense that it is the line that minimizes the distance between each original point and the line. If we drew a perpendicular line from each point to the regression line, and took the intersection of those lines as the approximation of the original datapoint, we would have a reduced representation of the original data that captures as much of the original variation as possible. Notice that there is a second regression line, perpendicular to the first, shown in Figure 2. This line captures as much of the variation as possible along the second dimension of the original data set. It does a poorer job of approximating the orginal data because it corresponds to a dimension exhibiting less variation to begin with. It is possible to use these regression lines to generate a set of uncorrelated data points that will show subgroupings in the original data not necessarily visible at first glance.

These are the basic ideas behind SVD: taking a high dimensional, highly variable set of data points and reducing it to a lower dimensional space that exposes the substructure of the

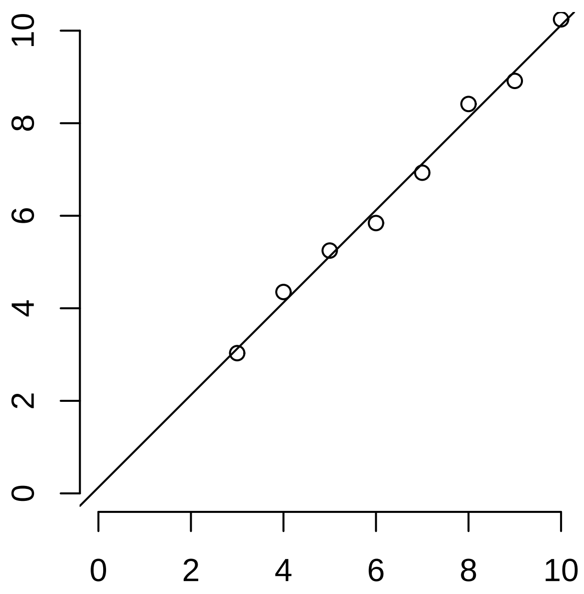


Figure 1: Best-fit regression line reduces data from two dimensions into one.

original data more clearly and orders it from most variation to the least. What makes SVD practical for NLP applications is that you can simply ignore variation below a particular threshhold to massively reduce your data but be assured that the main relationships of interest have been preserved.

8.1 Example of Full Singular Value Decomposition

SVD is based on a theorem from linear algebra which says that a rectangular matrix A can be broken down into the product of three matrices - an orthogonal matrix U, a diagonal matrix S, and the transpose of an orthogonal matrix V. The theorem is usually presented something like this:

$$A_{mn} = U_{mm} S_{mn} V_{nn}^T$$

where $U^TU = I, V^TV = I$; the columns of U are orthonormal eigenvectors of AA^T, the columns of V are orthonormal eigenvectors of A^TA, and S is a diagonal matrix containing the square roots of eigenvalues from U or V in descending order.

The following example merely applies this definition to a small matrix in order to compute its SVD. In the next section, I attempt to interpret the application of SVD to document classification.

Start with the matrix

$$A = \begin{bmatrix} 3 & 1 & 1 \\ -1 & 3 & 1 \end{bmatrix}$$

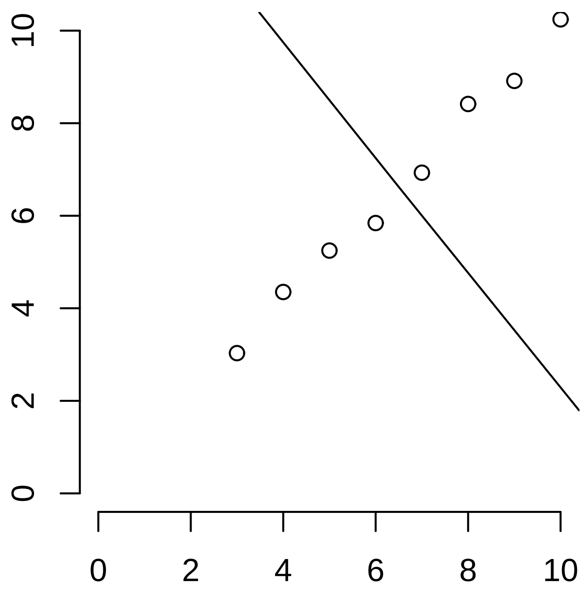


Figure 2: Regression line along second dimension captures less variation in original data.

In order to find U, we have to start with AA^T. The transpose of A is

$$A^T = \begin{bmatrix} 3 & -1 \\ 1 & 3 \\ 1 & 1 \end{bmatrix}$$

so

$$AA^T = \begin{bmatrix} 3 & 1 & 1 \\ -1 & 3 & 1 \end{bmatrix} \begin{bmatrix} 3 & -1 \\ 1 & 3 \\ 1 & 1 \end{bmatrix} = \begin{bmatrix} 11 & 1 \\ 1 & 11 \end{bmatrix}$$

Next, we have to find the eigenvalues and corresponding eigenvectors of AA^T. We know that eigenvectors are defined by the equation $A\vec{v} = \lambda\vec{v}$, and applying this to AA^T gives us

$$\begin{bmatrix} 11 & 1 \\ 1 & 11 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \lambda \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

We rewrite this as the set of equations

$$11x_1 + x_2 = \lambda x_1$$

$$x_1 + 11x_2 = \lambda x_2$$

and rearrange to get

$$(11 - \lambda)x_1 + x_2 = 0$$

$$x_1 + (11 - \lambda)x_2 = 0$$

Solve for λ by setting the determinant of the coefficient matrix to zero,

$$\begin{vmatrix} (11-\lambda) & 1 \\ 1 & (11-\lambda) \end{vmatrix} = 0$$

which works out as

$$(11-\lambda)(11-\lambda) - 1 \cdot 1 = 0$$

$$(\lambda - 10)(\lambda - 12) = 0$$

$$\lambda = 10, \lambda = 12$$

to give us our two eigenvalues $\lambda = 10, \lambda = 12$. Plugging λ back in to the original equations gives us our eigenvectors. For $\lambda = 10$ we get

$$(11-10)x_1 + x_2 = 0$$

$$x_1 = -x_2$$

which is true for lots of values, so we'll pick $x_1 = 1$ and $x_2 = -1$ since those are small and easier to work with. Thus, we have the eigenvector $[1, -1]$ corresponding to the eigenvalue $\lambda = 10$. For $\lambda = 12$ we have

$$(11-12)x_1 + x_2 = 0$$

$$x_1 = x_2$$

and for the same reason as before we'll take $x_1 = 1$ and $x_2 = 1$. Now, for $\lambda = 12$ we have the eigenvector $[1, 1]$. These eigenvectors become column vectors in a matrix ordered by the size of the corresponding eigenvalue. In other words, the eigenvector of the largest eigenvalue is column one, the eigenvector of the next largest eigenvalue is column two, and so forth and so on until we have the eigenvector of the smallest eigenvalue as the last column of our matrix. In the matrix below, the eigenvector for $\lambda = 12$ is column one, and the eigenvector for $\lambda = 10$ is column two.

$$\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

Finally, we have to convert this matrix into an orthogonal matrix which we do by applying the Gram-Schmidt orthonormalization process to the column vectors. Begin by normalizing $\vec{v_1}$.

$$\vec{u_1} = \frac{\vec{v_1}}{|\vec{v_1}|} = \frac{[1,1]}{\sqrt{1^2+1^2}} = \frac{[1,1]}{\sqrt{2}} = [\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}]$$

Compute

$$\vec{w_2} = \vec{v_2} - \vec{u_1} \cdot \vec{v_2} * \vec{u_1} =$$

$$[1,-1]-[\frac{1}{\sqrt{2}},\frac{1}{\sqrt{2}}]\cdot[1,-1]*[\frac{1}{\sqrt{2}},\frac{1}{\sqrt{2}}]=$$

$$[1,-1]-0*[\frac{1}{\sqrt{2}},\frac{1}{\sqrt{2}}]=[1,-1]-[0,0]=[1,-1]$$

and normalize

$$\vec{u_2}=\frac{\vec{w_2}}{|\vec{w_2}|}=[\frac{1}{\sqrt{2}},\frac{-1}{\sqrt{2}}]$$

to give

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

The calculation of V is similar. V is based on A^TA, so we have

$$A^TA=\begin{bmatrix}3 & -1\\ 1 & 3\\ 1 & 1\end{bmatrix}\begin{bmatrix}3 & 1 & 1\\ -1 & 3 & 1\end{bmatrix}=\begin{bmatrix}10 & 0 & 2\\ 0 & 10 & 4\\ 2 & 4 & 2\end{bmatrix}$$

Find the eigenvalues of A^TA by

$$\begin{bmatrix}10 & 0 & 2\\ 0 & 10 & 4\\ 2 & 4 & 2\end{bmatrix}\begin{bmatrix}x_1\\ x_2\\ x_3\end{bmatrix}=\lambda\begin{bmatrix}x_1\\ x_2\\ x_3\end{bmatrix}$$

which represents the system of equations

$$10x_1+2x_3=\lambda x_1$$

$$10x_2+4x_3=\lambda x_2$$

$$2x_1+4x_2+2x_3=\lambda x_2$$

which rewrite as

$$(10-\lambda)x_1+2x_3=0$$

$$(10-\lambda)x_2+4x_3=0$$

$$2x_1+4x_2+(2-\lambda)x_3=0$$

which are solved by setting

$$\begin{vmatrix}(10-\lambda) & 0 & 2\\ 0 & (10-\lambda) & 4\\ 2 & 4 & (2-\lambda)\end{vmatrix}=0$$

This works out as

$$(10-\lambda)\begin{vmatrix} (10-\lambda) & 4 \\ 4 & (2-\lambda) \end{vmatrix} + 2\begin{vmatrix} 0 & (10-\lambda) \\ 2 & 4 \end{vmatrix} =$$

$$(10-\lambda)[(10-\lambda)(2-\lambda)-16]+2[0-(20-2\lambda)] =$$

$$\lambda(\lambda-10)(\lambda-12)=0,$$

so $\lambda=0, \lambda=10, \lambda=12$ are the eigenvalues for A^TA. Substituting λ back into the original equations to find corresponding eigenvectors yields for $\lambda=12$

$$(10-12)x_1+2x_3=-2x_1+2x_3=0$$

$$x_1=1, x_3=1$$

$$(10-12)x_2+4x_3=-2x_2+4x_3=0$$

$$x_2=2x_3$$

$$x_2=2$$

So for $\lambda=12$, $\vec{v_1}=[1,2,1]$. For $\lambda=10$ we have

$$(10-10)x_1+2x_3=2x_3=0$$

$$x_3=0$$

$$2x_1+4x_2=0$$

$$x_1=-2x_2$$

$$x_1=2, x_2=-1$$

which means for $\lambda=10$, $\vec{v_2}=[2,-1,0]$. For $\lambda=0$ we have

$$10x_1+2x_3=0$$

$$x_3=-5$$

$$10x_1-20=0$$

$$x_2=2$$

$$2x_1+8-10=0$$

$$x_1=1$$

which means for $\lambda=0$, $\vec{v_3}=[1,2,-5]$. Order $\vec{v_1}$, $\vec{v_2}$, and $\vec{v_3}$ as column vectors in a matrix according to the size of the eigenvalue to get

$$\begin{bmatrix} 1 & 2 & 1 \\ 2 & -1 & 2 \\ 1 & 0 & -5 \end{bmatrix}$$

and use the Gram-Schmidt orthonormalization process to convert that to an orthonormal matrix.

$$\vec{u_1} = \frac{\vec{v_1}}{|\vec{v_1}|} = [\frac{1}{\sqrt{6}}, \frac{2}{\sqrt{6}}, \frac{1}{\sqrt{6}}]$$

$$\vec{w_2} = \vec{v_2} - \vec{u_1} \cdot \vec{v_2} * \vec{u_1} = [2, -1, 0]$$

$$\vec{u_2} = \frac{\vec{w_2}}{|\vec{w_2}|} = [\frac{2}{\sqrt{5}}, \frac{-1}{\sqrt{5}}, 0]$$

$$\vec{w_3} = \vec{v_3} - \vec{u_1} \cdot \vec{v_3} * \vec{u_1} - \vec{u_2} \cdot \vec{v_3} * \vec{u_2} = [\frac{-2}{3}, \frac{-4}{3}, \frac{10}{3}]$$

$$\vec{u_3} = \frac{\vec{w_3}}{|\vec{w_3}|} = [\frac{1}{\sqrt{30}}, \frac{2}{\sqrt{30}}, \frac{-5}{\sqrt{30}}]$$

All this to give us

$$V = \begin{bmatrix} \frac{1}{\sqrt{6}} & \frac{2}{\sqrt{5}} & \frac{1}{\sqrt{30}} \\ \frac{2}{\sqrt{6}} & \frac{-1}{\sqrt{5}} & \frac{2}{\sqrt{30}} \\ \frac{1}{\sqrt{6}} & 0 & \frac{-5}{\sqrt{30}} \end{bmatrix}$$

when we really want its transpose

$$V^T = \begin{bmatrix} \frac{1}{\sqrt{6}} & \frac{2}{\sqrt{6}} & \frac{1}{\sqrt{6}} \\ \frac{2}{\sqrt{5}} & \frac{-1}{\sqrt{5}} & 0 \\ \frac{1}{\sqrt{30}} & \frac{2}{\sqrt{30}} & \frac{-5}{\sqrt{30}} \end{bmatrix}$$

For S we take the square roots of the non-zero eigenvalues and populate the diagonal with them, putting the largest in s_{11}, the next largest in s_{22} and so on until the smallest value ends up in s_{mm}. The non-zero eigenvalues of U and V are always the same, so that's why it doesn't matter which one we take them from. Because we are doing full SVD, instead of reduced SVD (next section), we have to add a zero column vector to S so that it is of the proper dimensions to allow multiplication between U and V. The diagonal entries in S are the singular values of A, the columns in U are called left singular vectors, and the columns in V are called right singular vectors.

$$S = \begin{bmatrix} \sqrt{12} & 0 & 0 \\ 0 & \sqrt{10} & 0 \end{bmatrix}$$

Now we have all the pieces of the puzzle

$$A_{mn} = U_{mm} S_{mn} V_{nn}^T = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{-1}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} \sqrt{12} & 0 & 0 \\ 0 & \sqrt{10} & 0 \end{bmatrix} \begin{bmatrix} \frac{1}{\sqrt{6}} & \frac{2}{\sqrt{6}} & \frac{1}{\sqrt{6}} \\ \frac{2}{\sqrt{5}} & \frac{-1}{\sqrt{5}} & 0 \\ \frac{1}{\sqrt{30}} & \frac{2}{\sqrt{30}} & \frac{-5}{\sqrt{30}} \end{bmatrix} =$$

$$\begin{bmatrix} \frac{\sqrt{12}}{\sqrt{2}} & \frac{\sqrt{10}}{\sqrt{2}} & 0 \\ \frac{\sqrt{12}}{\sqrt{2}} & \frac{-\sqrt{10}}{\sqrt{2}} & 0 \end{bmatrix} \begin{bmatrix} \frac{1}{\sqrt{6}} & \frac{2}{\sqrt{6}} & \frac{1}{\sqrt{6}} \\ \frac{2}{\sqrt{5}} & \frac{-1}{\sqrt{5}} & 0 \\ \frac{1}{\sqrt{30}} & \frac{2}{\sqrt{30}} & \frac{-5}{\sqrt{30}} \end{bmatrix} = \begin{bmatrix} 3 & 1 & 1 \\ -1 & 3 & 1 \end{bmatrix}$$

8.2 Example of Reduced Singular Value Decomposition

Reduced singular value decomposition is the mathematical technique underlying a type of document retrieval and word similarity method variously called *Latent Semantic Indexing* or *Latent Semantic Analysis.* The insight underlying the use of SVD for these tasks is that it takes the original data, usually consisting of some variant of a word$\times$document matrix, and breaks it down into linearly independent components. These components are in some sense an abstraction away from the noisy correlations found in the original data to sets of values that best approximate the underlying structure of the dataset along each dimension independently. Because the majority of those components are very small, they can be ignored, resulting in an approximation of the data that contains substantially fewer dimensions than the original. SVD has the added benefit that in the process of dimensionality reduction, the representation of items that share substructure become more similar to each other, and items that were dissimilar to begin with may become more dissimilar as well. In practical terms, this means that documents about a particular topic become more similar even if the exact same words don't appear in all of them.

As we've already seen, SVD starts with a matrix, so we'll take the following word$\times$ document matrix as the starting point of the next example.

$$A = \begin{bmatrix} 2 & 0 & 8 & 6 & 0 \\ 1 & 6 & 0 & 1 & 7 \\ 5 & 0 & 7 & 4 & 0 \\ 7 & 0 & 8 & 5 & 0 \\ 0 & 10 & 0 & 0 & 7 \end{bmatrix}$$

Remember that to compute the SVD of a matrix A we want the product of three matrices such that

$$A = USV^T$$

where U and V are orthonormal and S is diagonal. The column vectors of U are taken from the orthonormal eigenvectors of AA^T, and ordered right to left from largest corresponding eigenvalue to the least. Notice that

$$AA^T = \begin{bmatrix} 2 & 0 & 8 & 6 & 0 \\ 1 & 6 & 0 & 1 & 7 \\ 5 & 0 & 7 & 4 & 0 \\ 7 & 0 & 8 & 5 & 0 \\ 0 & 10 & 0 & 0 & 7 \end{bmatrix} \begin{bmatrix} 2 & 1 & 5 & 7 & 0 \\ 0 & 6 & 0 & 0 & 10 \\ 8 & 0 & 7 & 8 & 0 \\ 6 & 1 & 4 & 5 & 0 \\ 0 & 7 & 0 & 0 & 7 \end{bmatrix} = \begin{bmatrix} 104 & 8 & 90 & 108 & 0 \\ 8 & 87 & 9 & 12 & 109 \\ 90 & 9 & 90 & 111 & 0 \\ 108 & 12 & 111 & 138 & 0 \\ 0 & 109 & 0 & 0 & 149 \end{bmatrix}$$

is a matrix whose values are the dot product of all the terms, so it is a kind of dispersion matrix of terms throughout all the documents. The singular values (eigenvalues) of AA^T are

$$\lambda = 321.07, \lambda = 230.17, \lambda = 12.70, \lambda = 3.94, \lambda = 0.12$$

which are used to compute and order the corresponding orthonormal singular vectors of U.

$$U = \begin{bmatrix} -0.54 & 0.07 & 0.82 & -0.11 & 0.12 \\ -0.10 & -0.59 & -0.11 & -0.79 & -0.06 \\ -0.53 & 0.06 & -0.21 & 0.12 & -0.81 \\ -0.65 & 0.07 & -0.51 & 0.06 & 0.56 \\ -0.06 & -0.80 & 0.09 & 0.59 & 0.04 \end{bmatrix}$$

This essentially gives a matrix in which words are represented as row vectors containing linearly independent components. Some word cooccurence patterns in these documents are indicated by the signs of the coefficients in U. For example, the signs in the first column vector are all negative, indicating the general cooccurence of words and documents. There are two groups visible in the second column vector of U: *car* and *wheel* have negative coefficients, while *doctor*, *nurse*, and *hospital* are all positive, indicating a grouping in which *wheel* only cooccurs with *car*. The third dimension indicates a grouping in which *car*, *nurse*, and *hospital* occur only with each other. The fourth dimension points out a pattern in which *nurse* and *hospital* occur in the absence of *wheel*, and the fifth dimension indicates a grouping in which *doctor* and *hospital* occur in the absence of *wheel*.

Computing V^T is similar. Since its values come from orthonormal singular vectors of A^TA, arranged right to left from largest corresponding singular value to the least, we have

$$A^TA = \begin{bmatrix} 79 & 6 & 107 & 68 & 7 \\ 6 & 136 & 0 & 6 & 112 \\ 107 & 0 & 177 & 116 & 0 \\ 68 & 6 & 116 & 78 & 7 \\ 7 & 112 & 0 & 7 & 98 \end{bmatrix}$$

which contains the dot product of all the documents. Applying the Gram-Schmidt orthonormalization process and taking the transpose yields

$$V^T = \begin{bmatrix} -0.46 & 0.02 & -0.87 & -0.00 & 0.17 \\ -0.07 & -0.76 & 0.06 & 0.60 & 0.23 \\ -0.74 & 0.10 & 0.28 & 0.22 & -0.56 \\ -0.48 & 0.03 & 0.40 & -0.33 & 0.70 \\ -0.07 & -0.64 & -0.04 & -0.69 & -0.32 \end{bmatrix}$$

S contains the square roots of the singular values ordered from greatest to least along its diagonal. These values indicate the variance of the linearly independent components along each dimension. In order to illustrate the effect of dimensionality reduction on this data set, we'll restrict S to the first three singular values to get

$$S = \begin{bmatrix} 17.92 & 0 & 0 \\ 0 & 15.17 & 0 \\ 0 & 0 & 3.56 \end{bmatrix}$$

In order for the matrix multiplication to go through, we have to eliminate the corresponding row vectors of U and corresponding column vectors of V^T to give us an approximation of A using 3 dimensions instead of the original 5. The result looks like this.

$$\hat{A} =$$

$$\begin{bmatrix} -0.54 & 0.07 & 0.82 \\ -0.10 & -0.59 & -0.11 \\ -0.53 & 0.06 & -0.21 \\ -0.65 & 0.07 & -0.51 \\ -0.06 & -0.80 & 0.09 \end{bmatrix} \begin{bmatrix} 17.92 & 0 & 0 \\ 0 & 15.17 & 0 \\ 0 & 0 & 3.56 \end{bmatrix} \begin{bmatrix} -0.46 & 0.02 & -0.87 & -0.00 & 0.17 \\ -0.07 & -0.76 & 0.06 & 0.60 & 0.23 \\ -0.74 & 0.10 & 0.28 & 0.22 & -0.56 \end{bmatrix}$$

$$= \begin{bmatrix} 2.29 & -0.66 & 9.33 & 1.25 & -3.09 \\ 1.77 & 6.76 & 0.90 & -5.50 & -2.13 \\ 4.86 & -0.96 & 8.01 & 0.38 & -0.97 \\ 6.62 & -1.23 & 9.58 & 0.24 & -0.71 \\ 1.14 & 9.19 & 0.33 & -7.19 & -3.13 \end{bmatrix}$$

In practice, however, the purpose is not to actually reconstruct the original matrix but to use the reduced dimensionality representation to identify similar words and documents. Documents are now represented by row vectors in V, and document similarity is obtained by comparing rows in the matrix VS (note that documents are represented as row vectors because we are working with V, not V^T). Words are represented by row vectors in U, and word similarity can be measured by computing row similarity in US.

Earlier I mentioned that in the process of dimensionality reduction, SVD makes similar items appear more similar, and unlike items more unlike. This can be explained by looking at the vectors in the reduced versions of U and V above. We know that the vectors contain components ordered from most to least amount of variation accounted for in the original data. By deleting elements representing dimensions which do not exhibit meaningful variation, we effectively eliminate noise in the representation of word vectors. Now the word vectors are shorter, and contain only the elements that account for the most significant correlations among words in the original dataset. The deleted elements had the effect of diluting these main correlations by introducing potential similarity along dimensions of questionable significance.

9 References

Deerwester, S., Dumais, S., Landauer, T., Furnas, G. and Harshman, R. (1990). "Indexing by Latent Semantic Analysis". Journal of the American Society of Information Science 41(6):391-407.

Ientilucci, E.J., (2003). "Using the Singular Value Decomposition". http://www.cis.rit.edu/~ejipci/research.htm

Jackson, J. E. (1991). *A User's Guide to Principal Components Analysis.* John Wiley & Sons, NY.

Manning, C. and Schütze, H. (1999). *Foundations of Statistical Natural Language Processing.* MIT Press, Cambridge, MA.

Marcus, M. and Minc, H. (1968). *Elementary Linear Algebra.* The MacMillan Company, NY.

perfectly