Questions

1. What is Singular Value Decomposition?

Decomposition of a matrix A into three matrices in the form $A = U\Sigma V^T$ is called singular value decomposition, where

A is a (m × n) matrix

U (m × m) and V^{T} $(\mathrm{n} \times \mathrm{n})$ are any two orthogonal matrices—not necessarily transposes of each other.

Σ (m × n) is a diagonal matrix.

2. What are singular values?

Eigenvalues of both AA^T and A^TA are the same. Singular values of A are the square roots of the nonzero eigenvalues of AA^T or A^TA denoted by $\sigma_{1,}\sigma_2, \ldots . \sigma_r$. There are r singular values.

3. How is the matrix U constructed?

U (m × m) is an Orthogonal matrix. The columns of the U matrix are called the left-singular vectors of A. The columns of *U* are formed from the eigenvectors of AA^T. We normalize the eigenvectors of AA^T to get an orthonormal set. If r < m, we have to extend the set to form an orthonormal basis. This can be dome by using Gram-Schmidt Process.

4. How is the matrix Σ constructed?

Σ (m × n) is a diagonal but rectangular matrix. The *r* singular values $\sigma_{1,}\sigma_2, \ldots . \sigma_r$ form the diagonal of Σ. All other elements of Σ other than the r singular values are zero. Matrix *A* has rank *r*.

$$\Sigma = \begin{bmatrix} D & 0 \\ 0 & 0 \end{bmatrix} \begin{matrix} \} \text{r} \\ \} \text{m} - \text{r} \end{matrix}$$

$$\underbrace{}_{\text{r}} \quad \underbrace{}_{\text{n} - \text{r}}$$

Each matrix O is a zero matrix of the appropriate size.

If m > n, then Σ is padded with extra rows of zeros in order to handle the extra rows in the matrix A. If m < n, then Σ is padded with extra columns of zeros in order to handle the extra rows in the matrix A.

D is an r × r diagonal matrix for some r not exceeding the smaller of m and n.

(If r equals m or n or both, some or all of the zero matrices do not appear.)

$$D = \begin{bmatrix} \sigma_1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \sigma_r \end{bmatrix}$$

5. How is the matrix V constructed?

V^{T} $(\mathrm{n} \times \mathrm{n})$ is an Orthogonal matrix. The columns of V are eigenvectors of A^TA. the columns of V are called the right-singular vectors of A.

6. Explain the relationship between SVD of a matrix and the four fundamental subspaces.
The matrix Σ has r singular values.
U(m × m) and V(n × n) give orthonormal bases for all four fundamental subspaces:

- First r columns of U $\{u_1, u_2, \ldots . u_r\}$ form the column space of A
- Last m−r columns of U $\{u_{r+1}, u_{r+2}, \ldots . u_m\}$ form the left nullspace of A
- First r columns of V $\{v_1, v_2, \ldots . v_r\}$ form the row space of A
- Last n−r columns of V $\{v_{r+1}, v_{r+2}, \ldots . v_m\}$ form the nullspace of A

7. Is the SVD of a matrix unique?
We can arrange eigenvectors in different orders to produce *U* and *V*. To standardize the solution, we order the eigenvectors such that vectors with higher eigenvalues come before those with smaller values. SVD is not unique.

8. What is padding in SVD?
If m > n, then Σ is padded with extra rows of zeros in order to handle the extra rows in the matrix A. If m < n, then Σ is padded with extra columns of zeros in order to handle the extra rows in the matrix A.

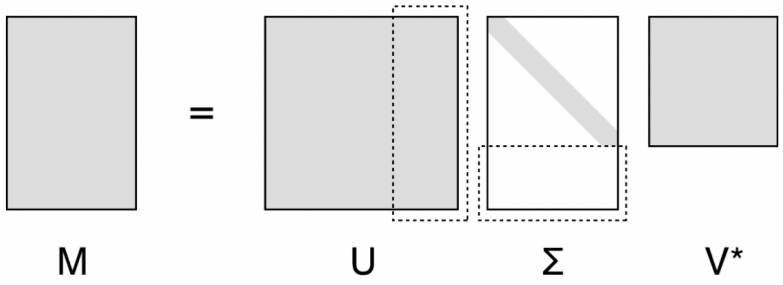


Figure 7: The canonical diagram of the SVD decomposition of a matrix M. The columns of U are the orthonormal left singular vectors; Σ is a diagonal matrix of singular values; and the rows of $V^\top$ are the orthonormal right singular vectors. The dashed areas are padding.

9. How can we determine the rank of matrix A?
The number of non-zero singular values is equal to the rank of A.

10. Which property of SVD is utilized for image compression or dimensionality reduction?
We order the eigenvectors such that vectors with higher eigenvalues come before those with smaller values. This turns out to be super-convenient when using SVD for applications like compression and dimensionality reduction, as you can simply choose the most "important" dimensions for the matrix representation as the first entries in the left or right singular vector matrices.

11. How can SVD be used for Latent semantic indexing?

One of first uses was in the field of information retrieval. The method that uses SVD is called latent semantic indexing (LSI) or latent semantic analysis.

In LSI, a matrix is constructed of documents and words. When the SVD is done on this matrix, it creates a set of singular values. The singular values represent concepts or topics contained in the documents. This was developed to allow more efficient searching of documents.

A simple search that looks only for the existence of words may have problems if the words are misspelled. Another problem with a simple search is that synonyms may be used, and looking for the existence of a word wouldn't tell you if a synonym was used to construct the document. If a concept is derived from thousands of similar documents, both of the synonyms will map to the same concept.

12. How can SVD be used for to compute Condition Number of a matrix?

Most numerical calculations involving an equation Ax = b are as reliable as possible when the SVD of A is used. The two orthogonal matrices U and V do not affect lengths of vectors or angles between vectors. Any possible instabilities in numerical calculations are identified in Σ. If the singular values of A are extremely large or small, roundoff errors are almost inevitable, but an error analysis is aided by knowing the entries in Σ and V .

If A is an invertible n×n matrix, then the ratio $\frac{\sigma_{max}}{\sigma_{min}}$ of the largest and smallest singular values gives the condition number of A.

The condition number affects the sensitivity of a solution of Ax = b to changes (or errors) in the entries of A. (Actually, a "condition number" of A can be computed in several ways, but the definition given here is widely used for studying Ax = b.)

Consider the following system of equations

$$(1 + 10^{-9})x + y = 1$$
$$x + (1 - 10^{-9})y = 1$$

Let $Let\ A = \begin{bmatrix} 1 + 10^{-9} & 1 \\ 1 & 1 - 10^{-9} \end{bmatrix}$

It is easily shown that the singular vectors are approximately $\begin{bmatrix} \sqrt{1/2} \\ \sqrt{1/2} \end{bmatrix}$ with singular value 2

and $\begin{bmatrix} \sqrt{1/2} \\ -\sqrt{1/2} \end{bmatrix}$ with singular value 10^{-9} .

Hence, the condition number is 2/ 10^{-9} = 2 · 10^{9}, so this matrix is very ill-conditioned. In solving the linear equation *Ax* = [1,1], note that the singular values of A^{-1} are 10^{9}, and 1/2, and the condition number is again 2 · 10^{9} .

13. How can we get a Truncated SVD?

Suppose that your data are encoded in an *m* × *n* matrix A . Compute the SVD decomposition $\mathrm{A} = U\Sigma V^T$. Choose a value *k* that is substantially less than *m* or *n*. Now let Σ_k be the k×k diagonal matrix with the first *k* singular values. Let V_k^T be the n×kmatrix whose first *k* rows are the same as V^T and the rest 0. Let U_k be the m×k matrix whose first *k* columns are the same as U and the rest 0.

Let A ≈ A_k= $U_k\ \Sigma_k V_k^T$.

A_k closely approximates A, particularly if the singular values that have been dropped are small compared to those that have been retained.

It is not difficult to see that the matrix A_k is of rank-k, and therefore it is viewed as a low-rank approximation of A .

Almost all forms of matrix factorization, including singular value decomposition, are low-rank approximations of the original matrix. Truncated singular value decomposition can retain a surprisingly large level of accuracy using values of k that are much smaller than min {m,n}. This is because only a very small proportion of the singular values are large in real world matrices. In such cases, A_k becomes an excellent approximation of A by retaining the few singular vectors that are large.

A useful property of truncated singular value decomposition is that it is also possible to create a lower dimensional representation of the data by changing the basis toV_k, so that each n-dimensional data point is now represented in only k dimensions. In other words, we change the axes so that the basis vectors correspond to the columns of V_k. This transformation is achieved by post-multiplying the data matrix A with $V_k$to obtain the m × k matrix P_k.

By post-multiplying A ≈ $U_k\Sigma_k V_k^T$ with V_k and using $V_k^T\ V_k = I_k$, we obtain the following:

P_k = A V_k = $U_k\ \Sigma_k$

Each row of P_k contains a reduced k-dimensional representation of the corresponding row in A. Therefore, we can obtain a reduced representation of the data either by post-multiplying the data matrix with the matrix containing the dominant right singular vectors (i.e., using A V_k), or we can simply scale the dominant left singular vectors with the singular values (i.e., using $U_k\ \Sigma_k$. Both these types of methods are used in real applications, depending on whether m or n is larger.

The reduction in dimensionality can be very significant in some domains such as images and text.

The matrix can be approximately reconstructed if the SVD is available.

How did we know how many singular values to keep? There are a number of heuristics for the number of singular values to keep. You typically want to keep 90% of the energy expressed in the matrix. To calculate the total energy, you add up all the squared singular values. You can then add squared singular values until you reach 90% of the total. Another heuristic to use when you have tens of thousands of singular values is to keep the first 2,000 or 3,000. This is a little less elegant than the energy method, but it's easier to implement in

practice. It's less elegant because you can't guarantee that 3,000 values contain 90% of the energy in any dataset.

14.? Given the SVD decomposition of the matrix: $\mathrm{A} = U\Sigma V^T$, how can the pseudo-inverse A^+ be written?

The Moore-Penrose pseudoinverse can be used for solving systems of linear equations, and for providing a solution to the problem of linear regression. SVD can be used to efficiently compute the Moore-Penrose pseudoinverse.

$$\mathrm{A} \approx U_k \Sigma_k V_k^T$$

This factorization of A is called Truncated Singular Value Decomposition
Since the diagonal entries in Σ are nonzero, we can form the following matrix, called the pseudoinverse (also, the Moore–Penrose inverse) of A:

$$A^+ = V_k \Sigma^{-1} U_k^T$$

15. How can SVD be used for Image Processing?
Suppose a satellite takes a picture, and wants to send it to Earth. The picture may contain 1000 by 1000 "pixels"—a million little squares, each with a definite color. We can code the colors, and send back 1,000,000 numbers. It is better to find the *essential* information inside the 1000 by 1000 matrix, and send only that.
Suppose we know the SVD. The key is in the singular values (in Σ). Typically, some σ's are significant and others are extremely small. If we keep 20 and throw away 980, then we send only the corresponding 20 columns of *U* and *V* . The other 980 columns are multiplied in *U* ΣV^T by the small σ 's that are being ignored.
We can do the matrix multiplication as columns times rows:

$$A = U\Sigma V^T = u_1\sigma_1 v_1^T + u_2\sigma_2 v_2^T + \ldots\ldots + u_r\sigma_r v_r^T$$

Any matrix is the sum of *r* matrices of rank 1. If only 20 terms are kept, we send 20 times 2000 numbers instead of a million (25 to 1 compression).
The pictures are really striking, as more and more singular values are included. At first you see nothing, and suddenly you recognize everything. The cost is in computing the SVD—this has become much more efficient, but it is expensive for a big matrix.

16. The SVD of a matrix was given as:

$$\mathrm{U} = \begin{bmatrix} -0.4346 & -0.3010 & 0.7745 & 0.3326 & -0.1000 \\ -0.1933 & -0.3934 & 0.1103 & -0.8886 & -0.0777 \\ 0.5484 & 0.5071 & 0.6045 & -0.2605 & -0.0944 \\ 0.6715 & -0.6841 & 0.0061 & 0.1770 & -0.2231 \\ 0.1488 & -0.1720 & 0.1502 & -0.0217 & 0.9619 \end{bmatrix}$$

$$\Sigma = \begin{bmatrix} 5.72 & 0 & 0 \\ 0 & 2.89 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

$$V = \begin{bmatrix} 0.2321 & -0.9483 & 0.2166 \\ -0.2770 & 0.1490 & 0.9493 \\ 0.9324 & 0.2803 & 0.2281 \end{bmatrix}$$

(a) Which columns form a basis for the null space of A? For the column space of A? For the row space of A?
(b) What are the singular values?
(c) What is the rank of A?
(d) What is the condition number of A? Is A ill-conditioned?

Solution

(a) The matrix A is 5 × 3, so the null space and row space are subspaces of R^3 . The column space (and null space of A^T) are in R^3 . The dimension of the column space (which is also the dimension of the row space) is 2.

Now we can answer the questions:

• The last column of V is a basis for the 1 - dimensional null space.
• The column space is spanned by the first two columns of U.
• The row space is spanned by the first two columns of V .

(b) $\sigma_1 = 5.72$, $\sigma_2 = 8.61$

(c) The rank is the dimension of the column space, which (given the SVD), is the number of non-zero singular values. In this case, the answer is 2.

(d) Condition number = $\frac{\sigma_{max}}{\sigma_{min}} = \frac{8.61}{5.72} = 1.5$

A is not ill-conditioned as the condition number is not very large.

17. What is Principal Component Analysis?

Consider for a moment the mass of data in figure 13.1. If I asked you to draw a line covering the data points, what's the longest possible line you could draw? I've drawn a few choices. Line B is the longest of these three lines. In PCA, we rotate the axes of the data. The rotation is determined by the data itself. The first axis is rotated to cover the largest variation in the data: line B in figure 13.1. The largest variation is the data telling us what's most important. It is the **first principal component.**

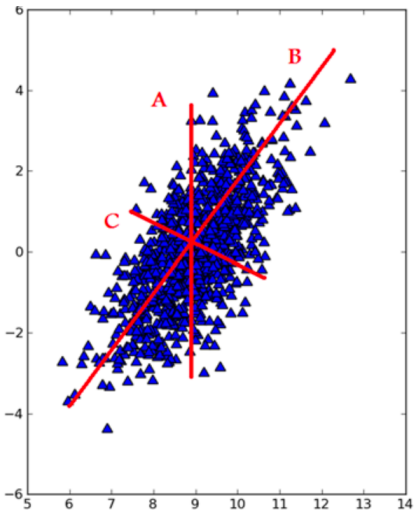


Figure 13.1 **Three choices for lines that span the entire dataset. Line B is the longest and accounts for the most variability in the dataset.**

After choosing the axis covering the most variability, we choose the next axis, which has the second most variability, provided it's perpendicular to the first axis. The real term used is orthogonal. On this two-dimensional plot, perpendicular and orthogonal are the same. In figure 13.1, line C would be our second axis. With PCA, we're rotating the axes so that they're lined up with the most important directions from the data's perspective.
We can get the values of the principal components by taking the covariance matrix of the data- set and doing eigenvalue analysis on the covariance matrix.
Once we have the eigenvectors of the covariance matrix, we can take the top N eigenvectors. The top N eigenvectors will give us the true structure of the N most important features. We can then multiply the data by the top N eigenvectors to transform our data into the new space.

18. What are Principal Components?
The unit eigenvectors $\mathbf{u}_1, \ldots, \mathbf{u}_p$ of the covariance matrix S are called the **principal components** of the data (in the matrix of observations). The **first principal component** is the eigenvector corresponding to the largest eigenvalue of S, the **second principal component** is the eigenvector corresponding to the second largest eigenvalue, and so on.

> The first principal component $\mathbf{u}_1$ determines the new variable y_1 in the following way. Let c1,...,cp be the entries in $\mathbf{u}_1$. Since $\mathbf{u}^T1$ is the first row of P^T , the equation $\mathbf{Y} = P^T \mathbf{X}$ shows that
> $y_1 = u_1^T X = c_1 x_1 + c_2 x_2 + \ldots\ldots\ldots c_p x_p$
> Thus y_1 is a linear combination of the original variables $x_1, \ldots, x_p$, using the entries in the eigenvector $\mathbf{u}_1$ as weights. In a similar fashion, $\mathbf{u}_2$ determines the variable y_2, and so on.

19. What do the covariances that we have as entries of the matrix tell us about the correlations between the observations?
Covariance is a measure for how well two data elements vary with each other
It's actually the sign of the covariance that matters :

- if positive then : the two observations increase or decrease together (correlated)
- if negative then : One increases when the other decreases (Inversely correlated)
- If zero: No relation between the observations

Now, that we know that the covariance matrix is not more than a table that summaries the correlations between all the possible pairs of observations.

20. How can PCA be used for image processing?

Typically, the image is 2000×2000 pixels, so there are 4 million pixels in the image. The data for the image form a matrix with 3 rows and 4 million columns (with columns arranged in any convenient order). Suppose, we would like to store the images using fewer pixels (or dimensions).

We make the image zero centered, ie subtract the average matrix elements from each of the matrix elements. Calculate the covariance matrix.

Calculate the Principal Components.

To get a low resolution image, we can store only the first few principal components.

We can remove the components with less information

21. How can PCA be used for dimensionality reduction?

Principal component analysis is potentially valuable for applications in which most of the variation, or dynamic range, in the data is due to variations in *only a few* of the new variables, $y_1, \ldots, y_p$.

It can be shown that an orthogonal change of variables, **X** = P **Y**, does not change the total variance of the data. (Roughly speaking, this is true because left-multiplication by P does not change the lengths of vectors or the angles between them.

This means that if S = PDP ^{T} , then

$$\begin{Bmatrix} Total\ Variance \\ of\ x_1, \ldots . x_p \end{Bmatrix} = \begin{Bmatrix} Total\ Variance \\ of\ y_1, \ldots . y_p \end{Bmatrix} = tr(D) = \lambda_1 + \ \ldots . + \lambda_p$$

The variance of y_j is λ_j , and the quotient λ_j / tr(S) measures the fraction of the total variance that is “explained” or “captured” by y_j .

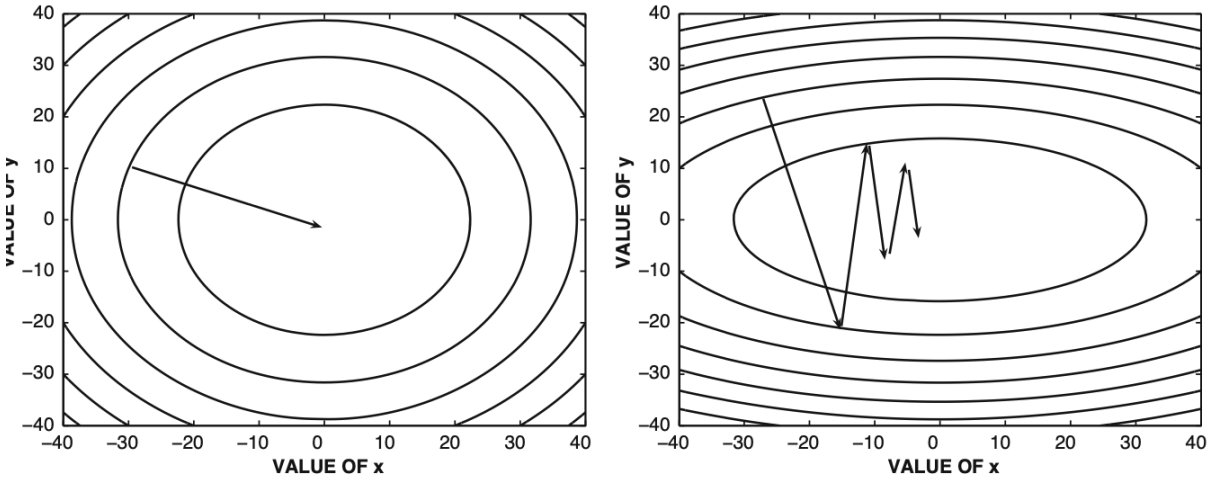
21. How can PCA be used for whitening in machine learning?

Principal component analysis is used for feature preprocessing in machine learning, by first reducing the dimensionality of the data and then normalizing the newly transformed features, so that the variance along each transformed direction is the same. Let V_k be the $d \times k$ matrix containing the top-k eigenvectors found by principal component analysis. Then, the first step is to transform the mean-centered data matrix D to the k-dimensional representation U_k as follows:

$$U_k = DV_k$$

- The next step is to divide each column of U_k by its standard deviation. As a result, the original data distribution becomes roughly spherical in shape. This type of approach is referred to as whitening.
- Standard deviation is how much variation the data has with respect to the mean

- This type of data distribution works much more effectively with gradient-descent algorithms. This is because widely varying variances along different directions also lead to loss functions in which different directions have different levels of curvature. Examples of two loss functions with different levels of curvature are illustrated in Figure 5.2. A loss function like Figure 5.2(a) tends to be more easily optimized with gradient descent algorithms.



(a) Loss function is circular bowl $L = x^2 + y^2$

(b) Loss function is elliptical bowl $L = x^2 + 4y^2$

Figure 5.2: The effect of the shape of the loss function on steepest-gradient descent

- Normalizing the data to have unit variance in all directions tends to reduce obvious forms of ill-conditioning in the loss function. As a result, gradient-descent tends to become much faster. Furthermore, the normalization of the data in this way sometimes prevents some subsets of features from having undue influence on the final results.

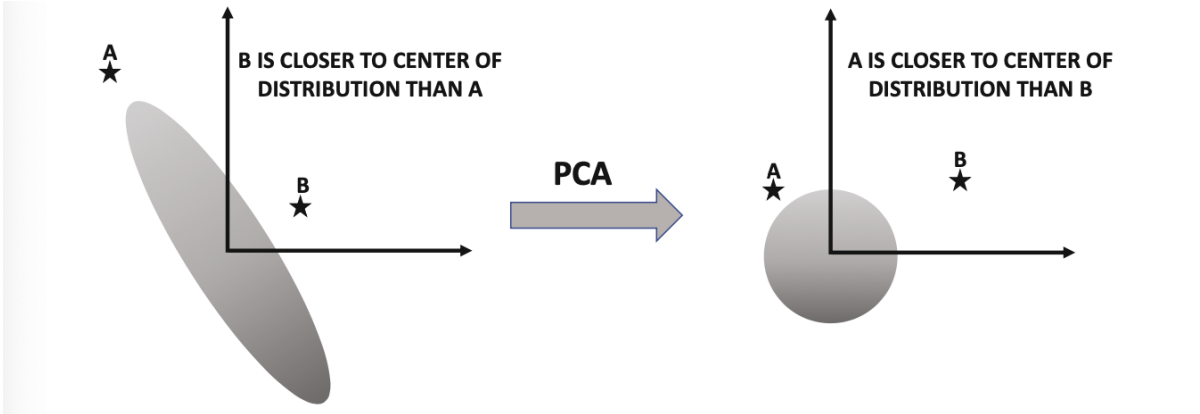


Figure 7.8: An example of the whitening of an ellipsoidal data distribution by principal component analysis and its use in outlier detection

This type of preprocessing is also used in unsupervised applications like outlier detection. In fact, whitening is arguably more important in unsupervised applications because one does not have labels to provide guidance about the relative importance of different directions in

the data. An example of the whitening of an ellipsoidal data distribution is illustrated in Figure 7.8. The resulting data distribution has a spherical shape.

21. How is PCA different from SVD?

The mean-centering of PCA helps in improving the accuracy of the approximation. In order to understand this point, we have shown an example of a 3-dimensional data set that is not originally mean-centered in Figure 7.6.

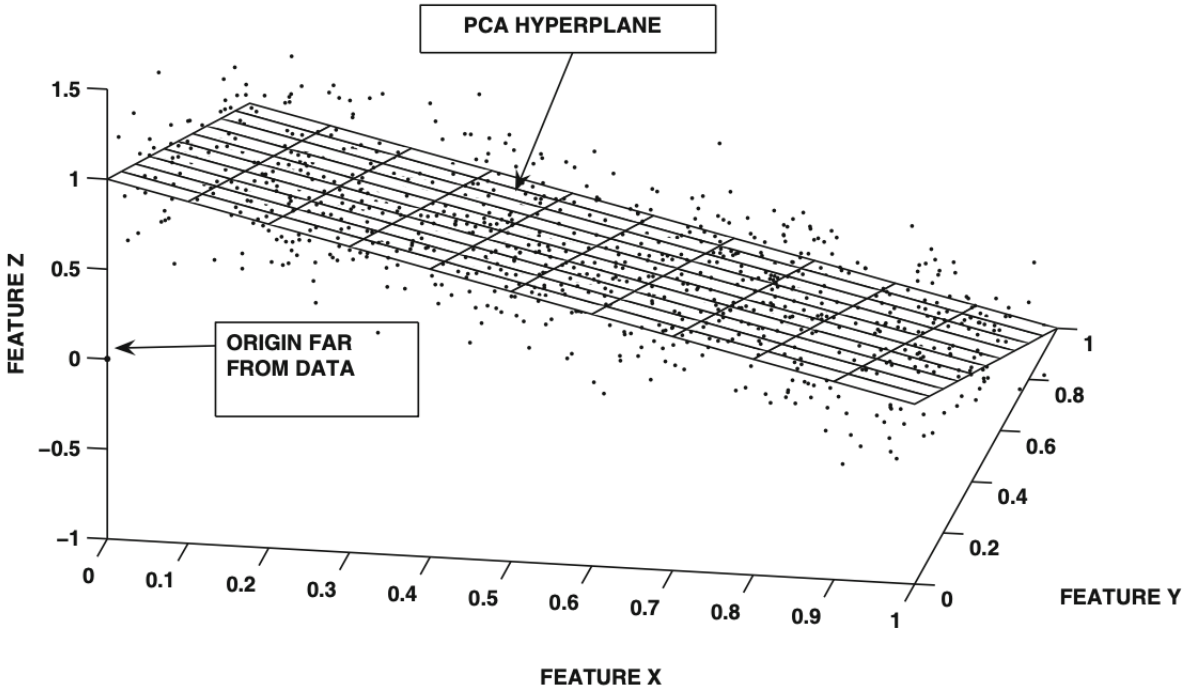


Figure 7.6: PCA for data that is not originally mean-centered

Most of the data is distributed near a plane far away from the origin (before preprocessing or mean-centering).

In this case, a 2-dimensional hyperplane can approximate the data quite well, where the mean-centering process ensures that the PCA hyperplane passed through the mean of the original data set. This is not the case for SVD, which will struggle to approximate the data without using all the three dimensions. It can be explicitly shown that the accuracy of PCA is at least as good as that of SVD for the same number of eigenvectors.