THE UNIVERSITY OF MINNESOTA

Statistics 5401 November 2, 2005

The Singular Value Decomposition and Some Applications

The singular value decomposition (SVD) of a matrix is a very useful tool in statistics.

Suppose $\mathbf{A} = [a_{ij}]$ is a n by p matrix where $n \geq p$. There is a mathematical theorem to the effect that $\mathbf{A}$ can always be represented (decomposed) as $\mathbf{A} = \mathbf{L}\,\mathbf{T}\,\mathbf{R}'$, where

$\mathbf{L}$ is n by p with orthonormal columns, that is, $\mathbf{L}'\mathbf{L} = I_p$,
$\mathbf{T} = \text{diag}[t_1, t_2, \ldots, t_p]$ is p by p diagonal, with $t_1 \geq t_2 \geq \ldots \geq t_p \geq 0$,
$\mathbf{R}$ is p by p with orthonormal columns, that is, $\mathbf{R}'\mathbf{R} = I_p$.

The columns of $\mathbf{L} = [\mathbf{L}_1,\ldots,\mathbf{L}_p]$ are the *left singular vectors* of $\mathbf{A}$. Each $\mathbf{L}_k$ is a vector of length n.

$t_1, t_2, \ldots, t_p$ are the *singular values* of $\mathbf{A}$ and are always assumed to be in decreasing order and non-negative.

The columns of $\mathbf{R} = [\mathbf{r}_1,\ldots,\mathbf{r}_p]$ are the *right singular vectors* of $\mathbf{A}$ Each $\mathbf{r}_k$ is a vector of length p.

The notation $\mathbf{L}$ and $\mathbf{R}$ stands for "left" and "right" and is not universally used.

Another way to express the SVD $\mathbf{A} = \mathbf{L}\,\mathbf{T}\,\mathbf{R}'$ is

$$\mathbf{A} = \sum_{1 \leq k \leq p} t_k \mathbf{L}_k \mathbf{r}_k',$$

a sum of <u>outer products</u> of left and right singular vectors weighted by singular values.

This representation makes clear that rank($\mathbf{A}$) = number of non-zero singular values. In fact, the best way to find the rank of a numerical matrix is usually to compute the singular values and count how many are essentially non-zero.

The SVD is related to eigenvalues and eigenvectors in at least two ways:

- The squared singular values $t_1^2 \geq t_2^2 \geq \ldots \geq t_p^2$ are the eigenvalues of the p by p symmetric matrix $\mathbf{A}'\mathbf{A}$ and the first p eigenvalues of the n by n symmetric matrix $\mathbf{A}\mathbf{A}'$.
- The right singular vectors $\mathbf{r}_1,\ldots,\mathbf{r}_p$ are the eigenvectors of $\mathbf{A}'\mathbf{A}$.
- The left singular vectors $\mathbf{L}_1,\ldots,\mathbf{L}_p$ are the eigenvectors of $\mathbf{A}\mathbf{A}'$ associated with the non-zero eigenvalues of $\mathbf{A}\mathbf{A}'$.

Although these properties suggest you could compute the SVD using software for computing eigenvalues and eigenvectors, it is better to use an algorithm specifically designed to compute the SVD, such as the one used by MacAnova function **svd()**. Here is a short example of its use.

A matrix **a** is first constructed, then decomposed.

```
Cmd> x <- run(10);a <- hconcat(rep(1,10),x,x^2,x^3); a
 (1,1)           1           1           1           1
 (2,1)           1           2           4           8
 (3,1)           1           3           9          27
 (4,1)           1           4          16          64
 (5,1)           1           5          25         125
 (6,1)           1           6          36         216
 (7,1)           1           7          49         343
 (8,1)           1           8          64         512
 (9,1)           1           9          81         729
(10,1)           1          10         100        1000

Cmd> svd(a) # just get singular values
(1)      1415.4       27.14      2.2961     0.41587
```

Since all 4 singular values are non-zero, the matrix has rank 4. However, since the smallest singular value is small relative to the largest, in a certain sense we can say the matrix is almost of rank 3.

```
Cmd> results <- svd(a, all:T);results # get everything
component: values
(1)      1415.4       27.14      2.2961     0.41587      vector
component: leftvectors
 (1,1) -0.00079055     0.04376    -0.48416    -0.76645
 (2,1)  -0.0059493     0.13086    -0.52913   -0.067025
 (3,1)    -0.01969      0.2395    -0.44324     0.26814
 (4,1)   -0.046225     0.34536     -0.2751     0.33222
 (5,1)   -0.089767     0.42413   -0.073307     0.21842   10 by 4
 (6,1)    -0.15453     0.45151     0.11353    0.019917
 (7,1)    -0.24472     0.40319     0.23679     -0.1701
 (8,1)    -0.36456     0.25487     0.24789    -0.25845
 (9,1)    -0.51825   -0.017766    0.098213    -0.15193
(10,1)    -0.71002    -0.43902    -0.26085     0.24263
component: rightvectors
(1,1)  -0.0015222    0.067665    -0.59637    -0.79985
(2,1)   -0.012736     0.27806    -0.75857     0.58914   4 by 4
(3,1)    -0.11096     0.95185     0.26184     -0.1145
(4,1)    -0.99374    -0.10995   -0.018601   0.0064589
```

Here are numerical checks of the properties of the SVD.

```
Cmd> L <- results$leftvectors; L' %*% L # left vectors orthonormal
(1,1)           1 -6.9389e-18 -1.1102e-16 -2.3592e-16
(2,1) -6.9389e-18           1  1.6653e-16  3.8858e-16
(3,1) -1.1102e-16  1.6653e-16           1 -4.4409e-16
(4,1) -2.3592e-16  3.8858e-16 -4.4409e-16           1
Cmd> R <- results$rightvectors; R' %*% R# right vectors orthonormal
(1,1)           1  4.0533e-17 -6.9218e-17  4.9043e-17
(2,1)  4.0533e-17           1  6.2095e-17  4.8526e-17
(3,1) -6.9218e-17  6.2095e-17           1 -1.4827e-16
(4,1)  4.9043e-17  4.8526e-17 -1.4827e-16           1
Cmd> L %*% dmat(results$values) %*% R' # reproduces matrix a
 (1,1)           1           1           1           1
 (2,1)           1           2           4           8
 (3,1)           1           3           9          27
 (4,1)           1           4          16          64
 (5,1)           1           5          25         125
 (6,1)           1           6          36         216
 (7,1)           1           7          49         343
 (8,1)           1           8          64         512
 (9,1)           1           9          81         729
(10,1)           1          10         100        1000
```

This confirms that you can reproduce a matrix from the three pieces of its SVD.

The SVD and low rank approximations to matrices

Suppose for any reason you would like to replace $\mathbf{A}$ by another n by p matrix, say $\hat{\mathbf{A}}_{(1)} = [\hat{a}_{ij}{}^{(1)}]$, which has rank 1 but, to some degree, approximates $\mathbf{A}$. How can you measure how good the approximation is. Statisticians think naturally of the sum of squares of the elements of the difference $\mathbf{A} - \hat{\mathbf{A}}_{(1)}$:

$$\|\mathbf{A} - \hat{\mathbf{A}}_{(1)}\|^2 \equiv \sum_i \sum_j (a_{ij} - \hat{a}_{ij}{}^{(1)})^2 = \mathrm{tr}((\mathbf{A} - \hat{\mathbf{A}}_{(1)})'(\mathbf{A} - \hat{\mathbf{A}}_{(1)}))$$

To say that $\hat{\mathbf{A}}_{(1)}$ has rank 1 means that $\hat{\mathbf{A}}_{(1)} = t\mathbf{L}\mathbf{r}'$, for some scalar t and normalized vectors $\mathbf{L}$ and $\mathbf{r}$ ($\mathbf{L}'\mathbf{L} = \mathbf{r}'\mathbf{r} = 1$).

Question: What rank 1 matrix $\hat{\mathbf{A}}_{(1)} = t\mathbf{L}\mathbf{r}'$ minimizes $\|\mathbf{A} - \hat{\mathbf{A}}_{(1)}\|^2$?

It is not difficult to show that the answer is $\hat{\mathbf{A}}_{(1)} = t_1\mathbf{L}_1\mathbf{r}_1'$, where t_1, $\mathbf{L}_1$, and $\mathbf{r}_1$ are the first singular value, and associated left and right singular vectors, respectively. Because of the minimizing property, we say $\hat{\mathbf{A}}_{(1)} = t_1\mathbf{L}_1\mathbf{r}_1'$ is the *least squares* rank 1 approximation to $\mathbf{A}$. If it is a good approximation, that is the residual sum of squares $\|\mathbf{A} - t_1\mathbf{L}_1\mathbf{r}_1'\|^2$ is small enough, you might say that $\mathbf{A}$ *almost* has rank 1.

When you generalize the question and seek for the best rank m > 1 approximation $\hat{\mathbf{A}}_{(m)}$ to $\mathbf{A}$ in the same least squares sense, the solution again comes from the SVD: The *least squares* rank m approximation to $\mathbf{A}$ is $\hat{\mathbf{A}}_{(m)} = \sum_{1 \le k \le m} \tau_k \boldsymbol{\lambda}_k \boldsymbol{\rho}_k'$, computed from the first m singular values, and right and left singular vectors. If $\|\mathbf{A} - \hat{\mathbf{A}}_{(m)}\|^2$ is quite small and $\|\mathbf{A} - \hat{\mathbf{A}}_{(m-1)}\|^2$ considerably larger, you might say that $\mathbf{A}$ *almost* has rank m.

The minimized sum of squared residuals is the sum of squares of the remaining singular values, $\tau_{m+1}, \ldots, \tau_p$:

$$\|\mathbf{A} - \hat{\mathbf{A}}_{(m)}\|^2 = \sum_i \sum_j (a_{ij} - \hat{a}_{ij(m)})^2 = \sum_{m+1 \le k \le p} \tau_k^2.$$

When this sum of the squares of the p-m smallest singular values is small relative to $\|\mathbf{A}\|^2 = \sum_i \sum_j a_{ij}^2 = \sum_{1 \le k \le p} \tau_k^2$, that is if

$$\sum_{m+1 \le k \le p} \tau_k^2 / \sum_{1 \le k \le p} \tau_k^2$$

is small, then the rank m approximation $\hat{\mathbf{A}}_{(m)}$ provides a good fit to $\mathbf{A}$.

Let's see how well the SVD-based approximations of different ranks actually approximate the matrix `a` in the example above. I used a `for(...){...}` loop to print out all 3 approximations in one command, omitting the rank 4 case which we have already seen exactly reproduces `a`.

```
Cmd> for(i,run(3)){ # loops over i from 1 to 3
       J <- run(i) # integers 1 to i to be used as subscript
       print(paste("Rank",i,"approximation"))
       results$leftvectors[,J] %*% dmat(results$values[J]) %*% \
       results$rightvectors[,J]'}
Rank 1 approximation
 (1,1)   0.0017032    0.014251     0.12416       1.112
 (2,1)    0.012818     0.10725     0.93436      8.3681
 (3,1)    0.042422     0.35494      3.0923      27.695
 (4,1)    0.099592     0.83328      7.2597      65.018
 (5,1)      0.1934      1.6182      14.098      126.26
 (6,1)     0.33293      2.7856      24.269      217.35
 (7,1)     0.52725      4.4115      38.434      344.21
 (8,1)     0.78544      6.5718      57.255      512.77
 (9,1)      1.1166      9.3424      81.393      728.95
(10,1)      1.5297      12.799      111.51      998.68
```

```
Rank 2 approximation
 (1,1)    0.082063     0.34448      1.2546     0.98138
 (2,1)     0.25314      1.0948      4.3149      7.9776
 (3,1)     0.48224      2.1623      9.2793       26.98
 (4,1)      0.7338      3.4394      16.181      63.987
 (5,1)     0.97227      4.8188      25.054         125
 (6,1)      1.1621      6.1929      35.933         216
 (7,1)      1.2677      7.4541       48.85      343.01
 (8,1)      1.2535      8.4951      63.839      512.01
 (9,1)       1.084      9.2083      80.934         729
(10,1)     0.72351      9.4862      100.17      999.99
Rank 3 approximation (very good approximation)
 (1,1)     0.74505      1.1878     0.96351      1.0021
 (2,1)     0.97771      2.0164      3.9968      8.0002
 (3,1)      1.0892      2.9343      9.0128      26.999
 (4,1)      1.1105      3.9186      16.016      63.999
 (5,1)      1.0727      4.9465       25.01         125
 (6,1)      1.0066      5.9951      36.001         216
 (7,1)     0.94342      7.0417      48.992         343
 (8,1)     0.91403      8.0633      63.988         512
 (9,1)     0.94946      9.0372      80.993         729
(10,1)      1.0807      9.9406      100.01        1000
```

As the rank increases, you get progressively better approximation. In particular, you get an excellent rank 3 approximation to **a**, justifying the claim that it is almost of rank 3.

Regression when both X and Y are measured with error

One application of this result is to the estimation of a simple linear relationship between random variables X and Y when *both* are observed subject to error (in ordinary regression, only Y is considered subject to error).

One way to formalize this is to assume that $X_i = x_i + \eta_i$ and $Y_i = y_i + \varepsilon_i$, where the x_i and y_i are not directly observable, but satisfy $y_i = \beta_0 + \beta_1 x_i$. η_i and ε_i are assumed independent measurement errors with $E[\eta_i] = E[\varepsilon_i] = 0$ and $V[\eta_i] = V[\varepsilon_i] = \sigma^2$. That is, the "true" values, x_i and y_i lie on a line in the x-y plane and the errors are independent with equal variances.

When X and Y are normal, it is not hard to show that the maximum likelihood estimate of the line passes through the point $(\overline{X}, \overline{Y})$, so that $\hat{\beta}_0 = \overline{Y} - \hat{\beta}_1 \overline{X}$, just as in ordinary least ordinary squares regression. Moreover, the best estimates of the remaining unknown quantities β_1, x_1, x_2, ..., x_N minimize the sum of the squared distances, measured perpendicular to the line, from the points (X_i, Y_i) to the points $(x_i, \overline{Y} + \beta_1(x_i - \overline{X}))$, all of which lie on the line.

This sum of squared perpendicular distances is

$$RSS = \sum_i (X_i - \overline{X} - (x_i - \overline{X}))^2 + \sum_i (Y_i - \overline{Y} - \beta_1 (x_i - \overline{X}))^2$$

Note that RSS includes square X-residuals as well as Y-residuals.

Define $\mathbf{A} = [\mathbf{X} - \overline{X}\mathbf{1}_N, \mathbf{Y} - \overline{Y}\mathbf{1}_N]$ and $\tilde{\mathbf{A}} = (\mathbf{x} - \overline{X}\mathbf{1}_N)[1 \quad \beta_1]$, a matrix with rank 1. Here $\mathbf{X} = [X_i]$ and $\mathbf{Y} = [Y_i]$ are n by 1 column vectors of the observed data and $\mathbf{x} = [x_i]$ is a possible vector of the *unobserved* "true" x's. $\mathbf{A}$ and $\tilde{\mathbf{A}}$ are both N by 2 and $RSS = \|\mathbf{A} - \tilde{\mathbf{A}}\|^2$.

Therefore, if $\mathbf{A} = \mathbf{L}\mathbf{T}\mathbf{R}'$ is the SVD of $[\mathbf{X} - \overline{X}\mathbf{1}_n, \mathbf{Y} - \overline{Y}\mathbf{1}_n]$, the $\tilde{\mathbf{A}}$ which minimizes RSS is $\hat{\mathbf{A}}_{(1)} = t_1 \mathbf{L}_1 \mathbf{r}_1'$.

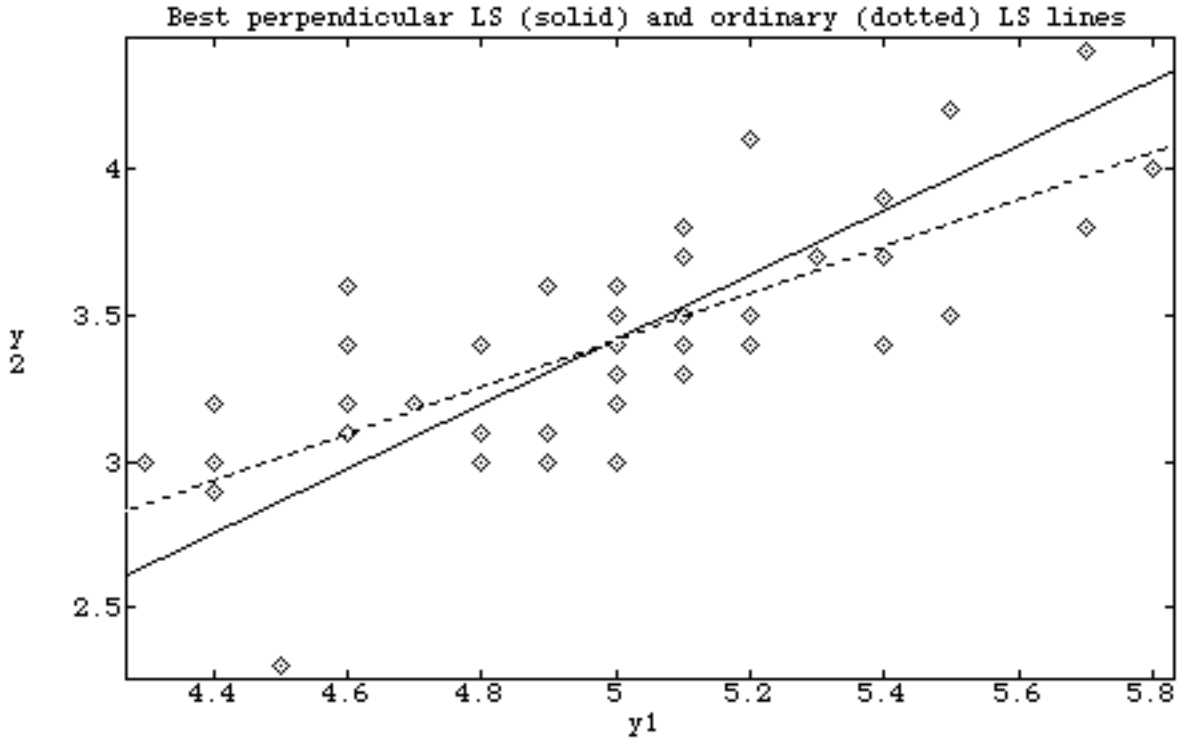
It follows that $[1 \quad \hat{\beta}_1]$ is proportional to $\mathbf{r}_1' = [r_{11}, r_{21}]$ and thus $\hat{\beta}_1 = r_{21}/r_{11}$. Moreover, if you are interested in it, you can estimate the true $\mathbf{x}$ by $\hat{\mathbf{x}} = \overline{X}\mathbf{1}_n + s_1 r_{11} \mathbf{L}_1$.

Here is an illustration of this usage, using data on the first two variables, extracted as **y1** and **y2**, for variety I. Setosa in the Fisher Iris data.

```
Cmd> data <- read("","t11_05",quiet:T) # read JWData5.txt
Read from file "TP1:Stat5401:Data:JWData5.txt"

Cmd> groups <- factor(data[,1]); y <- data[,-1]

Cmd> y1 <- y[groups == 1,1]; y2 <- y[groups == 1,2]

Cmd> means <- describe(hconcat(y1,y2),mean:T);means
(1)       5.006       3.428

Cmd> svdstuff <- svd(hconcat(y1-means[1],y2-means[2]),all:T)

Cmd> s1 <- svdstuff$values[1] # first (largest) singular value

Cmd> r1 <- vector(svdstuff$rightvectors[,1])# its right sing. vector

Cmd> l1 <- vector(svdstuff$leftvectors[,1])# its left sing. vector

Cmd> beta1 <- r1[2]/r1[1]; beta0 <- means[2] - beta1*means[1]

Cmd> vector(beta0, beta1)
(1)     -2.0924      1.1028

Cmd> # Here is ordinary LS regression

Cmd> regress("y2 = y1")
Model used is y2 = y1
                Coef      StdErr           t
CONSTANT    -0.56943     0.52171     -1.0915
y1           0.79853     0.10397      7.6807

N: 50, MSE: 0.065806, DF: 48, R^2: 0.55138
Regression F(1,48): 58.994, Durbin-Watson: 2.3454
To see the ANOVA table type 'anova()'
```

Note that the least squares regression coefficients are quite different from those determined from the SVD. So let's compare the two lines.

```
Cmd> plot(y1, y2, symbols:"\1",show:F) # sstart graph with points
Cmd> x0 <- vector(4,6) # x-values at ends of lines to be drawn
Cmd> addlines(x0, beta0+beta1* x0,show:F) # added line from ASVD
Cmd> addlines(x0, COEF[1] + COEF[2]*x0, linetype:3,\
 xlab:"y1", ylab:"y2",\
 title:"Best perpendicular LS (solid) and ordinary (dotted) LS lines")
```



Remark: Because the scale of the two axes are not identical, the distances whose squares are minimized are not actually the perpendicular distances in the graph from the line. They would be if the scales were identical.

You can generalize this result to p variables. When you have sample of N multivariate p-dimensional random vectors $\mathbf{x}_1,\ldots,\mathbf{x}_N$, whose expectations are assumed to lie on a line in p-dimensional space, and $\mathbf{X} = [\mathbf{x}_1,\ldots,\mathbf{x}_N]'$ is the data matrix, then the best fitting line in the perpendicular least square sense runs through the sample mean $\overline{\mathbf{x}} = N^{-1}\sum\mathbf{x}_i$ and and has the same direction as $\mathbf{L}_1$, the first right singular vector of the residuals $\tilde{\mathbf{X}} = \mathbf{X} - 1_N\overline{\mathbf{x}}'$ from the mean. Essentially you are finding a rank one approximation to $\tilde{\mathbf{X}}$. Alternatively, you can view $1_N\overline{\mathbf{x}}' + t_1\mathbf{L}_1\mathbf{r}_1'$ as a rank 2 approximation to $\mathbf{X}$, with the first component selected on a priori grounds and the second computed from the configuration of the data points. It is not hard to show that $1_n'\mathbf{L}_j = 0$ for the

left singular vectors $\mathbf{L}_j$ of X.

SVD and Principal components

The preceding, in which the matrix $\tilde{\mathbf{X}} = \mathbf{X} - \mathbf{1}_N\overline{\mathbf{x}}'$ is approximated by a rank 1 matrix, suggests it may be interesting to approximate $\tilde{\mathbf{X}}$ by a rank m matrix $\hat{\tilde{\mathbf{X}}}_{(m)}$, where $1 \le m < p$. Then you could approximate the original data matrix by the rank m+1 matrix $\mathbf{1}_N\overline{\mathbf{x}}' + \hat{\tilde{\mathbf{X}}}_{(m)}$. From the general results for the SVD, the best (in the least squares sense) choice of $\hat{\tilde{\mathbf{X}}}_{(m)}$ is

$$\hat{\tilde{\mathbf{X}}}_{(m)} = \sum_{1\le k\le m} t_k \mathbf{L}_k \mathbf{r}_k',$$

where the t_k, $\mathbf{L}_k$, and $\mathbf{r}_k$ are the singular values and vectors of $\tilde{\mathbf{X}}$. When the approximation is a good one, that is $\sum_{m+1\le k\le p} t_k^2 / \sum_{1\le k\le p} t_k^2$ is small, for some purposes it may be possible to replace your original p variables, the columns of $\mathbf{X}$, by m new variables, defined by the n by 1 left singular vectors $\mathbf{L}_1$, ..., $\mathbf{L}_m$. If you also retain the sample mean $\overline{\mathbf{x}}$ and the first m right singular vectors $\mathbf{r}_1$, ..., $\mathbf{r}_m$, you can almost recover $\mathbf{X}$ and hence little information may have been lost. This results in an effective reduction in dimensionality from p to m < p.

In fact, the left singular vectors are closely related to the *principal components* of the data. Specifically, let $\mathbf{u}_k$ be the k^{th} normalized eigenvector of the sample covariance matrix $\hat{V}[\mathbf{x}] = (N-1)^{-1}\sum(\mathbf{x}_i - \overline{\mathbf{x}})(\mathbf{x}_i - \overline{\mathbf{x}})' = (N-1)^{-1}\tilde{\mathbf{X}}'\tilde{\mathbf{X}}$. Then you can express $\mathbf{Z}_k = \mathbf{X}\mathbf{u}_k$, the vector containing the values of the k^{th} principal component, as $\mathbf{Z}_k = \mathbf{1}_N(\overline{\mathbf{x}}'\mathbf{u}_k) + t_k\mathbf{L}_k$.

This is completely expressible in terms of the SVD because $\mathbf{u}_k = \mathbf{r}_k$, where $\mathbf{r}_k$ is the k^{th} right singular vector so that $\mathbf{Z}_k = \mathbf{1}_N(\overline{\mathbf{x}}'\mathbf{r}_k) + t_k\mathbf{L}_k$. Thus except for a scaling factor and the addition of scalar $\overline{\mathbf{x}}'\mathbf{r}_k$, the principal components are the same as the left singular vectors of $\tilde{\mathbf{X}}$. The matrix of all principal components is $\mathbf{Z} = \mathbf{X}\mathbf{U} = \mathbf{1}_N(\overline{\mathbf{x}}'\mathbf{R}) + \mathbf{L}\mathbf{T}$, if $\mathbf{U} = [\mathbf{u}_1,\ldots,\mathbf{u}_p]$.

We illustrate this again with the Fisher Iris data.

```
Cmd> stats <- tabs(y,covar:T,mean:T)
Cmd> compnames(stats)
(1) "mean"
(2) "covar"
Cmd> ybar <- stats$mean; sy <- stats$covar; N <- nrows(y)
Cmd> svdstuff <- svd(y - ybar',all:T)
Cmd> compnames(svdstuff)
(1) "values"
(2) "leftvectors"
(3) "rightvectors"
```

```
Cmd> eigs <- eigen(sy) # sqrt of (N-1)*eigenvalues = singular valUEs
Cmd> hconcat(sqrt((N-1)*eigs$values), svdstuff$values)
(1,1)         25.1         25.1
(2,1)       6.0131       6.0131
(3,1)       3.4137       3.4137
(4,1)       1.8845       1.8845
```

Since $\mathbf{S} = (N-1)^{-1}\tilde{\mathbf{X}}'\tilde{\mathbf{X}}$, the elements of `(N-1)*eigs$values` are the eigenvalues of $\tilde{\mathbf{X}}'\tilde{\mathbf{X}}$, the squares of the singular values of $\tilde{\mathbf{X}}$.

```
Cmd> zc <- y %*% eigs$vectors # compute covariance princ. components
Cmd> tvals <- svdstuff$values # singular values
Cmd> left <- svdstuff$leftvectors # left singular vectors
Cmd> right <- svdstuff$rightvectors # right singular vectors
Cmd> ones <- rep(1,N)
Cmd> zc1 <- ones %*% (ybar' %*% right) + left %*% dmat(tvals)
Cmd> zc1 [run(5),] # 1st 5 case of PC's computed from SVD
             (1)          (2)          (3)          (4)
(1)       2.8182      -5.6463      0.65977     0.031089
(2)       2.7882        -5.15      0.84232    -0.065675
(3)       2.6134       -5.182      0.61395     0.013383
(4)        2.757      -5.0087      0.60029      0.10893
(5)       2.7736      -5.6537      0.54177      0.09461
Cmd> zc[run(5),] # first 5 cases of PC's computed from S
             (1)          (2)          (3)          (4)
(1)       2.8182       5.6463     -0.65977     0.031089
(2)       2.7882         5.15     -0.84232    -0.065675
(3)       2.6134        5.182     -0.61395     0.013383
(4)        2.757       5.0087     -0.60029      0.10893
(5)       2.7736       5.6537     -0.54177      0.09461
```

The two ways of computing principal components yield the same answers except for signs.