PCA and eigendecomposition

March 30, 2021

1 PCA and Eigendecomposition

1.1 Steps

1. Generate bi-variate normal random numbers
2. Perform PCA on the bi-variate normal random numbers
3. Perform PCA step-by-step
4. Perform eigendecomposition step-by-step

1.2 Generate bi-variate normal random numbers

Generate random numbers from the same multivariate normal distribution. Define μ and Cov, and generate 100 random numbers.

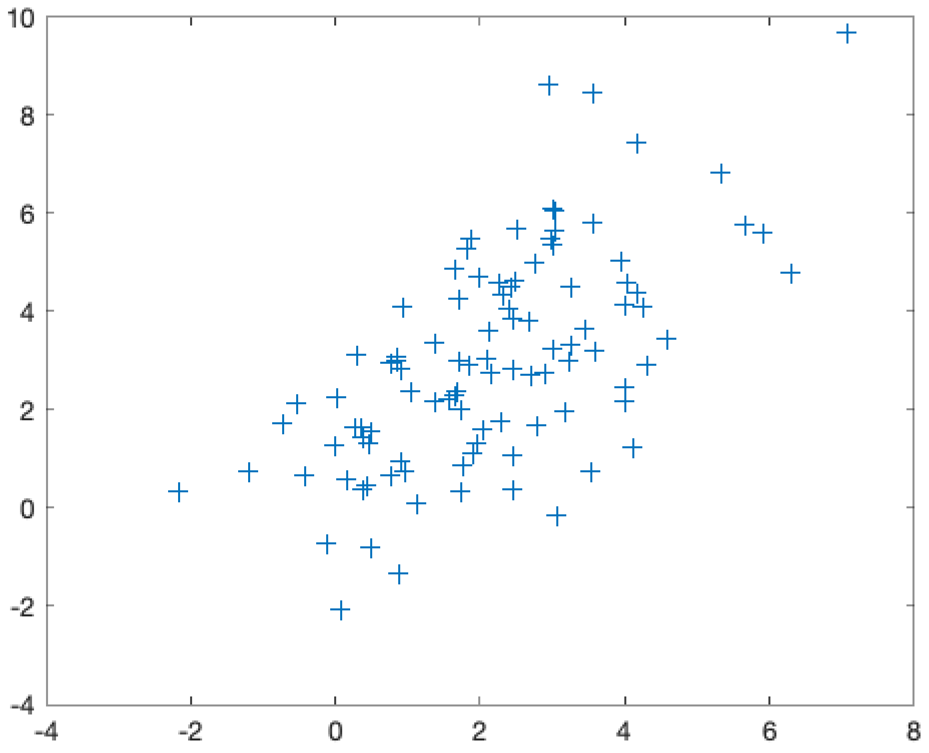
Covariances of multivariate normal distributions, specified as a d-by-d (here $d = 2$) symmetric, positive semi-definite matrix Cov, the covariance matrix. The covariance matrix is diagonal, containing variances along the diagonal and the covariance off it.

[1]:
```
format compact % For compact printing of results
mu = [2 3];
Cov = [2 1.5; 1.5 4]
rng('default')  % For reproducibility
R = mvnrnd(mu,Cov,100);
```

```
Cov =
    2.0000    1.5000
    1.5000    4.0000
```

Plot the random numbers.

[2]:
```
plot(R(:,1),R(:,2),'+');
```



Check sample variance and covariance of the generated data points and the sample correlation between the variables.

```
[3]: Sample_cov = cov(R)
     Cor = corrcoef(R)
```

```
Sample_cov =
    2.7023    2.2380
    2.2380    4.7411
Cor =
    1.0000    0.6253
    0.6253    1.0000
```

Covariance and variance:

$$Cov(X,Y) = \frac{1}{n}\sum_{i=1}^{n}((x_i - \mu_X)(y_i - \mu_Y))$$

$$Var(X) = Cov(X,X) = \sigma_X^2$$

Pearson's coefficient of correlation

$$Cor(X,Y) = \frac{Cov(X,Y)}{\sigma_X \sigma_Y} = \frac{Cov(X,Y)}{\sqrt{Var(X)}\sqrt{Var(Y)}}$$

In our example $Cov(X,Y) = 2.1031$, $Var(X) = 1.3512$, and $Var(Y) = 4.0267$. As expected:

[4]:
```
r = 2.2380/(sqrt(2.7023)*sqrt(4.7411))
```

```
r =
    0.6253
```

1.3 Perform PCA on the bi-variate normal random numbers

[5]:
```
[loading_vector,score,latent,tsquared,variablity_explained,mu] = pca(R);
loading_vector
variablity_explained
```

```
loading_vector =
    0.5411    0.8410
    0.8410   -0.5411
variablity_explained =
   83.0393
   16.9607
```

Check the correlation of the scores. As expected:

[6]:
```
corrcoef(score)
```

```
ans =
    1.0000   -0.0000
   -0.0000    1.0000
```

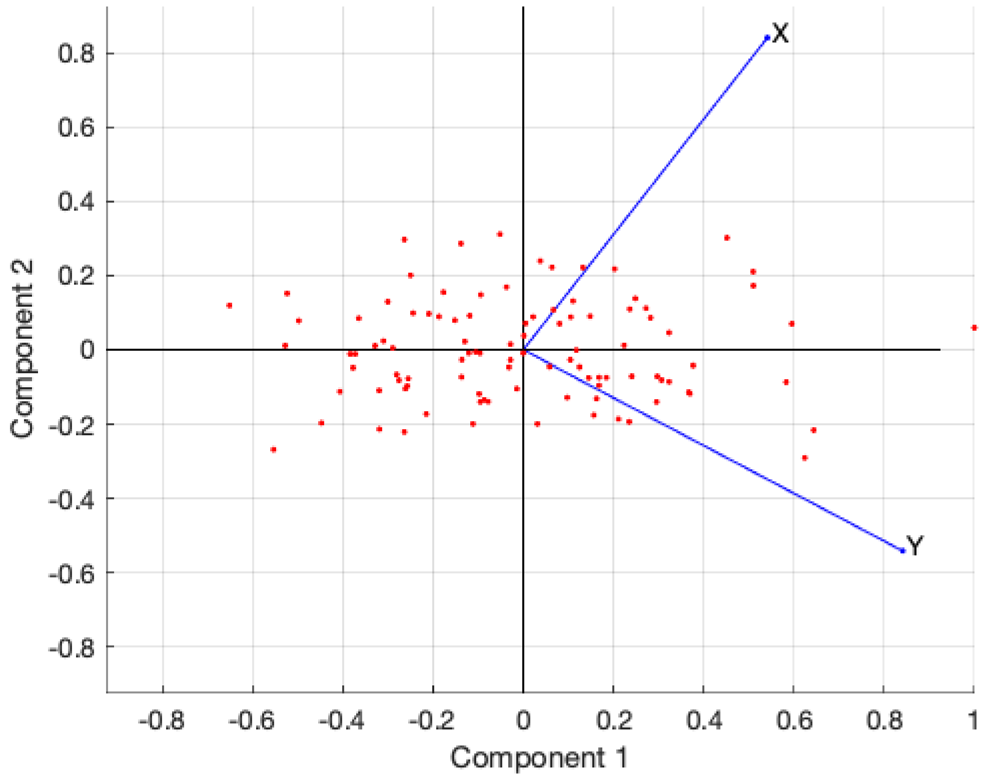
The eigenvectors are unit vectors and orthogonal, therefore the 2-Norm is 1 and the inner (scalar, dot) product is 0.

[7]:
```
norm(loading_vector(:,1))
norm(loading_vector(:,2))
dot(loading_vector(:,1),loading_vector(:,2))
```

```
ans =
     1
ans =
     1
ans =
     0
```

Produce a biplot of the scores and loading vectors of the two principle components.

[8]:
```
biplot(loading_vector,'scores',score,'varlabels',{'X','Y'});
```



1.4 Perform PCA step-by-step

1. Center and Standardize, i.e., subtracting the mean of the data from the original dataset
2. Find the covariance matrix of the dataset
3. Find the eigenvectors associated with the greatest eigenvalues of the covariance matrix
4. Project the original dataset on the eigenvectors

1.4.1 Standardize

Compute standard scores (commonly called z-scores) defined as $z_i = \frac{x_i - \mu_X}{\sigma_X}$.

[9]:
```
data(:,1) = (R(:,1)-mean(R(:,1)))/sqrt(2.7023);
data(:,2) = (R(:,2)-mean(R(:,2)))/sqrt(4.7411);
```

1.4.2 Find the covariance matrix

As before. Check out the effect of standardization on the covariance matrix.

[10]:
```
Cov = cov(data)
```

```
Cov =
    1.0000    0.6253
    0.6253    1.0000
```

1.4.3 Find the eigenvectors and eigenvalues

Recall the definition of eigenvectors v and -values λ: $Av = \lambda v$, where A is an n-by-n matrix, v is a column vector of length n, and λ is a scalar.

[11]:
```
[loading_vector, lambda] = eig(Cov)
```

```
loading_vector =
    0.7071   -0.7071
   -0.7071   -0.7071
lambda =
    0.3747         0
         0    1.6253
```

Mind that the second eigenvalue of 1.6253 is larger than the first eigenvalue of 0.3747.

Lets check if $Cov\ v = \lambda v$ for the two eigenvectors and -values. As expected:

[12]:
```
Cov*loading_vector(:,1) - lambda(1,1)*loading_vector(:,1)
Cov*loading_vector(:,2) - lambda(2,2)*loading_vector(:,2)
```

```
ans =
   1.0e-16 *
   -0.5551
         0
ans =
   1.0e-15 *
         0
   -0.2220
```

The eigenvectors returned are unit vectors and orthogonal, therefore the 2-Norm is 1 and the inner (scalar, dot) product is 0.

[13]:
```
norm(loading_vector(:,1))
norm(loading_vector(:,2))
dot(loading_vector(:,1),loading_vector(:,2))
```

```
ans =
     1
ans =
     1
ans =
     0
```

1.4.4 Project the original dataset on the eigenvectors

Compute the scores of the data points corresponding to the two principle components. Since the second eigenvalue of 2.6326 is larger than the first eigenvalue of 0.5448, we need to flip the scores left/right since the second column is first principle component (the one with the largest eigenvalue).

In `Matlab`, the operation `A'` computes the transpose A^T of a matrix A.

[14]:
```
score = loading_vector * data';
score = score';
score = fliplr(score);
```

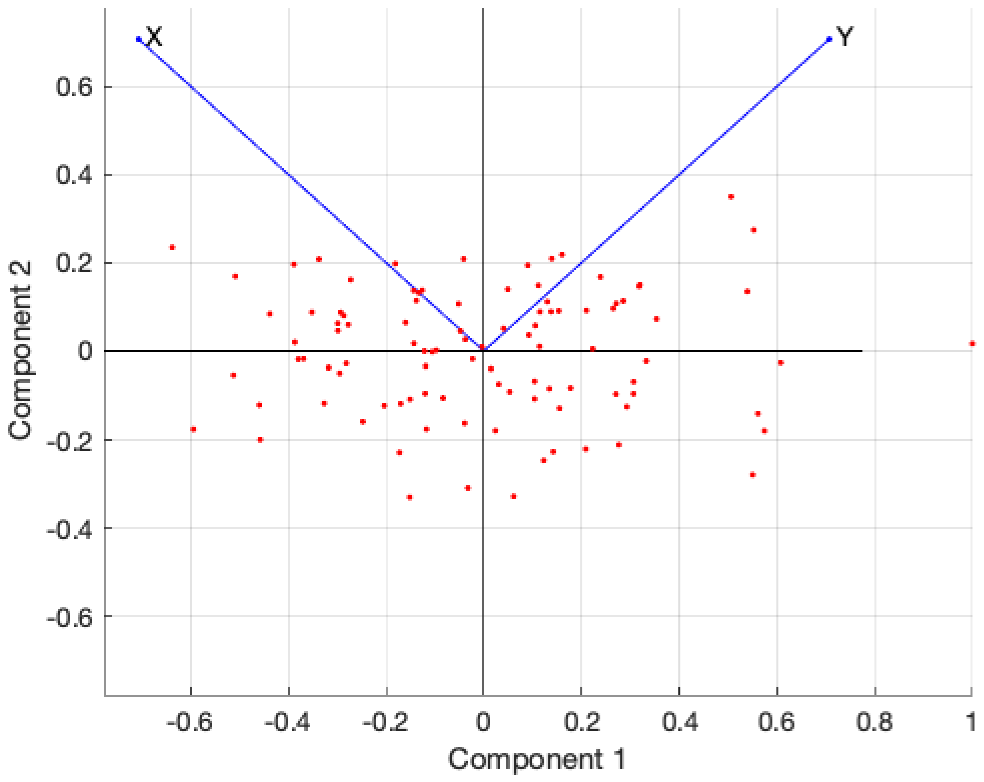
Check the correlation of the scores. As expected:

[15]:
```
corrcoef(score)
```

```
ans =
    1.0000   -0.0000
   -0.0000    1.0000
```

Produce a biplot of the scores and loading vectors of the two principle components.

[16]:
```
biplot(loading_vector,'scores',score,'varlabels',{'X','Y'});
```



The eigenvalues of the covariance matrix indicate the variance in this (new) coordinate direction. We can use this information to calculate the relative variance explained by each component: for each component, we divide the variances according to the eigenvectors by the sum of the variances. In our example, the 1st and 2nd principal component contains 81.26 and 16.53 percent of the total variance in data.

[22]:
```
var(score)
var(score)/sum(var(score))
```

```
ans =
    1.6253    0.3747
ans =
    0.8126    0.1874
```

Why does this look differently compared to the plot when using the `pca` *library? Fix the problem!*

1.5 Perform eigendecomposition step-by-step

1.5.1 Eigenvalues

We find the eigenvalues of an $n \times n$ (covariance) matrix C by solving the so-called characteristic equation $\det(C – \lambda I) = 0$ where I is the identity matrix of size n.

Here $C = Cov$ is a 2×2 matrix and the determinant of such a matrix is defiend as $\det(C) = c_{11}c_{22} - c_{12}c_{21}$. Hence, we need to solve:

$$\begin{aligned} \det(Cov – \lambda I) &= (Cov_{11} - \lambda)(Cov_{22} - \lambda) - Cov_{12}Cov_{21} & (1) \\ &= \lambda^2 - (Cov_{11} + Cov_{22})\lambda + Cov_{11}Cov_{22} - Cov_{12}Cov_{21} & (2) \\ &= 0 & (3) \end{aligned}$$

It's a quadriatic equation in λ with, in general, two solutions.

[25]:
```
Cov
a = 1;
b = -Cov(1,1)-Cov(2,2);
c = Cov(1,1)*Cov(2,2)-Cov(1,2)*Cov(2,1);

polynomial = [a b c];
eigenvalues = roots(polynomial)
```

```
Cov =
    1.0000    0.6253
    0.6253    1.0000
eigenvalues =
    1.6253
    0.3747
```

Let's test the two solutions λ_1 and λ_2. As expected:

[26]:
```
lambda1 = diag([eigenvalues(1) eigenvalues(1)])
Char1 = Cov - lambda1;
det(Char1)
lambda2 = diag([eigenvalues(2) eigenvalues(2)]);
Char2 = Cov - lambda2;
det(Char2)
```

```
lambda1 =
    1.6253         0
         0    1.6253
ans =
  -2.3893e-16
ans =
  -1.1314e-16
```

BTW, in `Matlab` we can write the shortcut using the characteristic polynomial `poly` of a matrix :

[27]:
```
eigenvalues = roots(poly(Cov))
eigenvalues =
    1.6253
    0.3747
```

1.5.2 Eigenvectors

All that's left is to find the two eigenvectors v_1, v_2. We understand that by definition:

$$
\begin{align}
Cv &= \lambda v \tag{4}\\
Cv - \lambda v &= 0 \tag{5}\\
Cv - \lambda I v &= 0 \tag{6}\\
(C - \lambda I)v &= 0 \tag{7}\\
& \tag{8}
\end{align}
$$

This is nothing but solving a linear equation system.

Before, we compute the eigenvector v_1 associated with the eigenvalue $\lambda_1 = 1.6253$, we first, check if a solution exits. Therfore, we comput the rank, the determinat, and the reduced row echelon form of the matrix $A_1 = C – \lambda_1 I$.

In `Matlab` the function `rref` computes the reduced row echelon form of a matrix resulting from a Gaussian elimination.

[28]:
```
A1 = Cov - lambda1
A1_rank = rank(A1)
A1_det = det(A1)
A1_red = rref(A1)
```

```
A1 =
   -0.6253    0.6253
    0.6253   -0.6253
A1_rank =
     1
A1_det =
  -2.3893e-16
A1_red =
    1.0000   -1.0000
         0         0
```

As we see, we have a *general* problem: the matrix A_1 is not fully ranked, hence, the linear equation system is underdetermined, i.e. we have infinitly many solution. Consequently, its deteminat is 0 and A_1 is, hence, not invertible.

However, the length and the orientation of an eigenvector v are arbitrary. So we can set the last entry to anything, say $v_1(2) = 1$, compute $v_1(1)$, and then fix the length of v to get a unit vector.

Let $_1$ be in reduced row echelon form, i.e. $A_1(i,i) = 1$, and $v_1(2) = 1$. The linear equation reduces to

$$\begin{aligned}
A_1(1,1)v_1(1) + A_1(1,2)v_1(2) &= 0 && (9)\\
1v_1(1) + A_1(1,2)1 &= 0 && (10)\\
v_1(1) + A_1(1,2) &= 0 && (11)\\
v_1(1) &= -A_1(1,2) && (12)\\
& && (13)
\end{aligned}$$

To make it a unit length vector, we divide by the length (2-norm) of the vector.

[30]:
```
v1 = [-A1_red(1,2);1];
eigenvector1 = v1/norm(v1)
```

```
eigenvector1 =
    0.7071
    0.7071
```

Finally, we are ready to compute the eigenvector v_2 associated with the eigenvalue $\lambda_2 = 0.3747$ in the same way.

[31]:
```
A2 = Cov - lambda2
A2_red = rref(A2)
v2 = [-A2_red(1,2);1];
eigenvector2 = v2/norm(v2)
```

```
A2 =
    0.6253    0.6253
    0.6253    0.6253
A2_red =
    1.0000    1.0000
         0         0
eigenvector2 =
   -0.7071
    0.7071
```

1.6 References

1. http://mres.uni-potsdam.de/index.php/2017/09/14/principal-component-analysis-in-6-steps/
2. https://www.youtube.com/watch?v=ssfMqFycXOU