# Lecture 12
# Principal Component Analysis Demonstrations

Jason J. Bramburger

---

This lecture does not come from a chapter in the textbook, but serves as a nice introduction to Section 15.3 on principal component analysis. The hope is that this demonstration will help you to better understand the theory that we will be presented with in the following lecture. Hence, some of you will be tempted to ask "why?" sometimes during this lecture. Your questions will be answered in full by the end of next lecture.

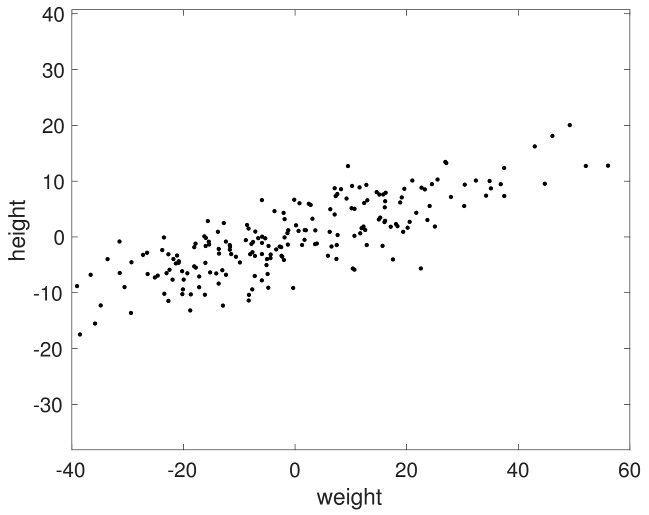
We saw in the previous lecture that the SVD can be used to produce low-rank approximations of a data set. In this lecture (and the one that follows) we will see what these low-rank approximations mean geometrically when we are working with data matrices. This method is called **principal component analysis**.

## Weight and Height Data

To get started, we will use a data set that has the weights and heights of 200 people. These data are contained in the file 'weightheight.mat'. The weights are in the first row and the heights are contained in the second row. We note that the average weight and average height have been subtracted off so that each row has zero mean. Let's go ahead and plot the data:

```
load('weightheight.mat')

plot(X(1,:),X(2,:),'k.','MarkerSize',10)
hold on
axis equal
xlabel('weight')
ylabel('height')
set(gca,'Fontsize',16)
```



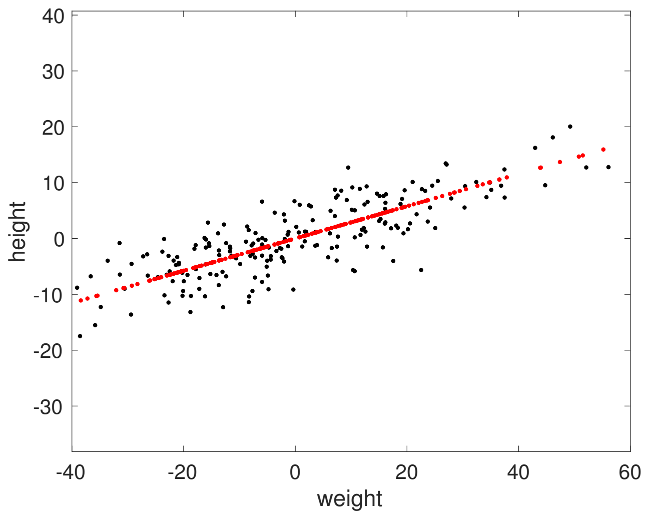
Just on inspection, it seems that there is a slight correlation between weight and height here. We are going to use the SVD to better understand this. Since we only want the SVD for low-rank approximations, there is no use in obtaining the full SVD which includes a number of zero rows padded into the $\Sigma$ matrix (denoted $S$ in the code). We use 'econ' in MATLAB's built-in svd() function to indicate this.

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

The matrix $U$ is a $2 \times 2$ matrix which has the left singular vectors as columns, the matrix $S$ is a $2 \times 2$ diagonal matrix with the singular values on the diagonal, and the matrix $V$ is a $200 \times 2$ matrix which has the right singular vectors as columns. Be careful: this command gives you $V$ and not $V^*$, so multiplying $U$, $S$, and $V$ together will not give you $X$.

Let's calculate and plot the rank-1 approximation of the matrix $X$:

```
X_rank1 = S(1,1)*U(:,1)*V(:,1)';
plot(X_rank1(1,:),X_rank1(2,:),'r.','MarkerSize',10)
```



It is immediately apparent that all the points in the rank-1 approximation all fall on a single line. If you think of the data being two-dimension (weight and height), the rank-1 approximation represents a 1-dimensional approximation of the data. Each red point in the rank-1 approximation corresponds to one of the black data points. In fact, these red points are projections of the full data onto a line. Geometrically, to get the position of the red point, you trace the shortest path from the data point to the line on which all the red points fall. That path will be perpendicular to the line.

```
plot([X(1,10) X_rank1(1,10)],[X(2,10) X_rank1(2,10)],'color',[0 0.5 0],'Linewidth',2)
```

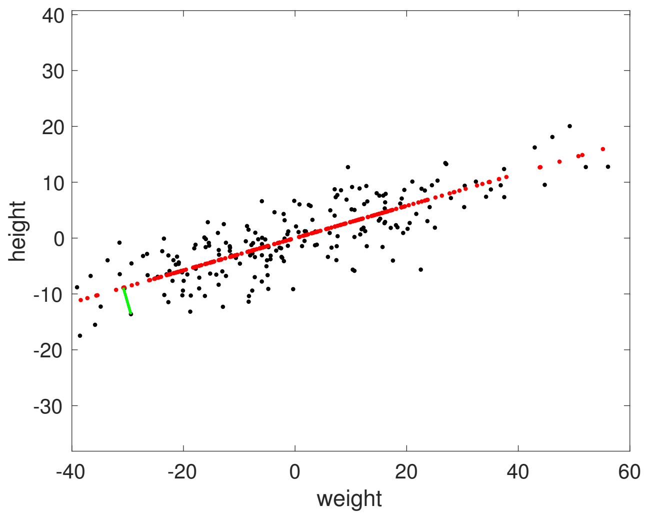
Using the theorem from the end of Lecture 11, we have that our X_rank1 exactly minimizes the sum of squares of these perpendicular distances. Recall that these distances are quantified by the Frobenius norm, and our theorem gave us that the minimizer is exactly the rank-1 approximation of our data matrix.

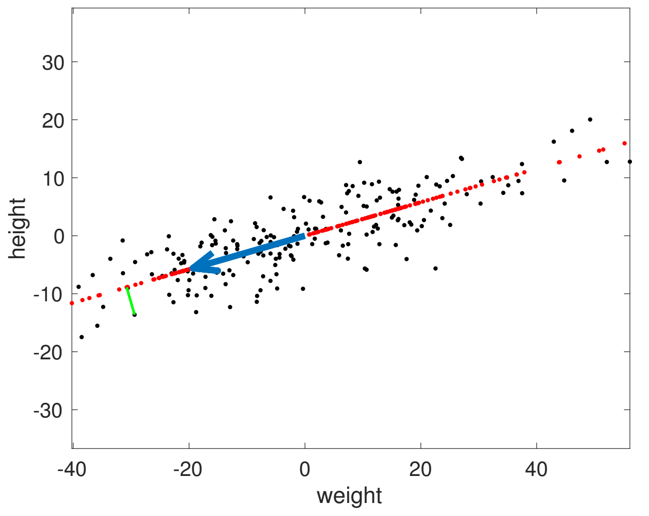
Let's calculate and plot the vector $y = (\sigma_1/\sqrt{199})u_1$. The 199 is not arbitrary, it is actually $n - 1$, where $n$ is the number of columns in our data matrix.

```
n = 200;
y1 = S(1,1)/sqrt(n-1)*U(:,1);
c = compass(y1(1),y1(2)); % creates vector from the origin pointed at y1
set(c,'Linewidth',4)
```

The vector falls exactly on the line of the rank-1 approximation! So, the direction of the vector $u_1$ defines the line on which the rank-1 approximation falls. This is the direction on which the data varies the most.

Let's do the same thing with $u_2$:





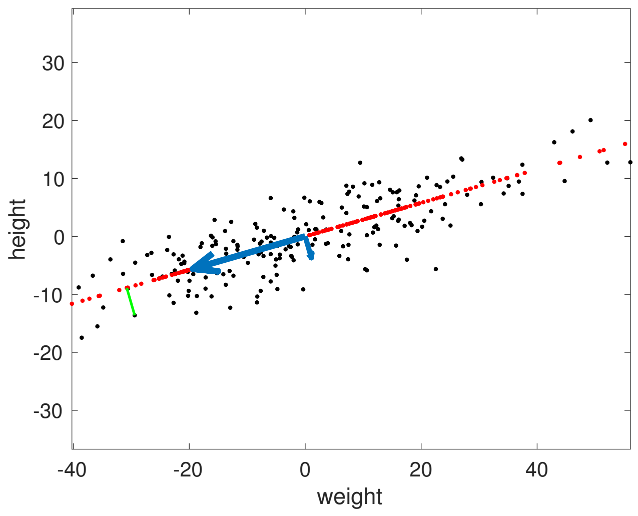
```
y2 = S(2,2)/sqrt(n-1)*U(:,2);
c = compass(y2(1),y2(2));
set(c,'Linewidth',4)
```

There are two things that you should take note of here. First, the vectors in the directions of $u_1$ and $u_2$ are perpendicular. Although it might be surprising to see for the first time, this is to be expected since $U$ is an orthogonal matrix. This means that the columns of $U$ are orthogonal/perpendicular vectors. Second, the vector in the direction of $u_2$ is shorter than the vector in the direction of $u_1$. This is because the lengths of the vectors are proportional to the spread of the data in that direction. Precisely, the lengths of these vectors are proportional to the standard deviation of the data in that direction.

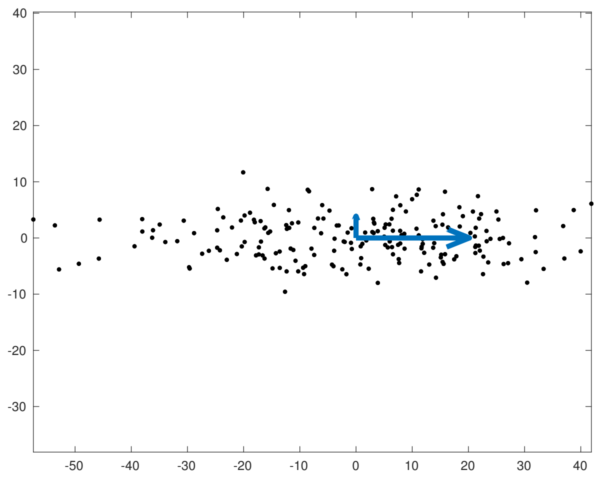
The two perpendicular vectors plotted above are called the *principal components* of the data matrix $X$. We say that $u_1$ is the first principal component, while $u_2$ is the second. Since these vectors are perpendicular, it is often best to work in a basis of the principal components. In our present example, this would mean that we don't describe data points in terms of heights and weights, but in terms of their positions relative to $u_1$ and $u_2$. To convert to this principal component basis, we can just multiply the data by $U'$ on the left.

```
X_proj = U'*X;

figure(2)
plot(X_proj(1,:),X_proj(2,:),'k.','MarkerSize',10)
```



```
axis equal
hold on
y1_proj = U'*y1;
y2_proj = U'*y2;
c = compass(y1_proj(1),y1_proj(2));
set(c,'Linewidth',4);
c = compass(y2_proj(1),y2_proj(2));
set(c,'Linewidth',4);
```



In this new basis, the perpendicular directions are uncorrelated. That is, in the heigh-weight plot, generally if you increased one you also increased the other. Now, in this new basis, knowing something about the $x$-value tells you nothing about the $y$-value.

## A Three-Dimensional Example

The principals laid out in our weight and height data example extend to arbitrarily high dimensions. Let's look at a 3D dataset and see what the low-rank approximations look like.

```
% Clean workspace
clear all; close all; clc

rng(5); % make random numbers be the same every time
n = 200; % number of data points
```

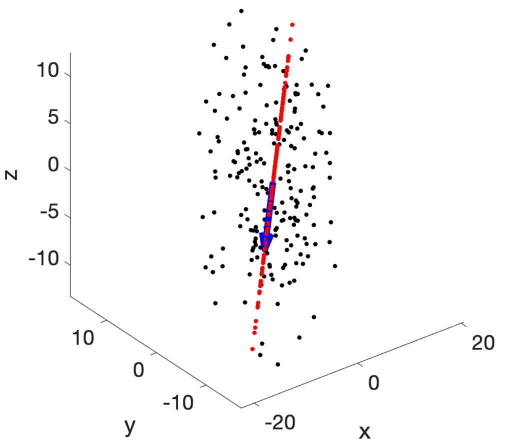
```
X = [3 6 4; 6 3 0; 4 0 1]*randn(3,n); % create n data points in 3D

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

figure(3)
plot3(X(1,:),X(2,:),X(3,:),'k.','Markersize',10)
axis vis3d
hold on
xlabel('x')
ylabel('y')
zlabel('z')
set(gca,'Fontsize',16)

% Compute the rank-1 approximation of the data
X_rank1 = U(:,1)*S(1,1)*V(:,1)';
plot3(X_rank1(1,:),X_rank1(2,:),X_rank1(3,:),'r.','Markersize',10)

% Plot the vector sqrt(sigma_1)*u_1/(n-1)
vec = S(1,1)/sqrt(n-1) * U(:,1);
quiver3(0,0,0, vec(1),vec(2),vec(3),0,'b','Linewidth',5,'maxheadsize',1)
```



Just like before, the rank-1 approximation falls on a line. Furthermore, that line is generated by the vector $(\sigma_1/\sqrt{199})u_1$. Again, this line is the direction on which the data varies the most.

Let's do the same thing for the vector generated by $u_2$:

```
vec = S(2,2)/sqrt(n-1) * U(:,2);
quiver3(0,0,0, vec(1),vec(2),vec(3),0,'b','Linewidth',5,'maxheadsize',1)
```

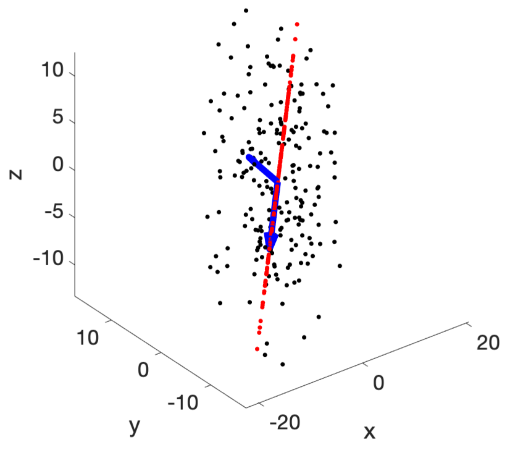
Finally, let's compute the rank-2 approximation and add it to the plot:

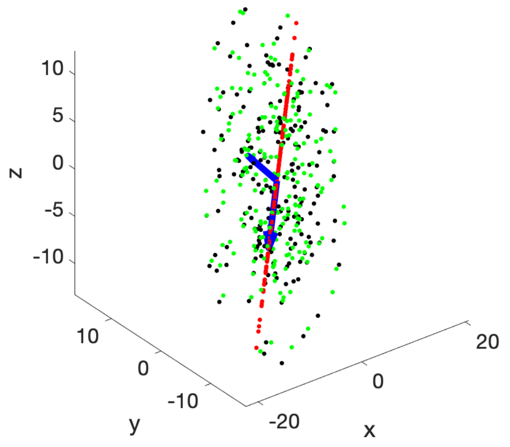
```
X_rank2 = U(:,1)*S(1,1)*V(:,1)' + U(:,2)*S(2,2)*V(:,2)';
plot3(X_rank2(1,:),X_rank2(2,:),X_rank2(3,:),'g.','Markersize',10)
```

The rank-2 approximation lies along a plane spanned by $u_1$ and $u_2$. It is also the plane that minimizes the squared distances between the data points and any possible plane. Again, this property is a consequence of the theorem from the end of Lecture 11 with $N = 2$.

## Another 3D Example

To finish, let's consider one more 3D dataset.





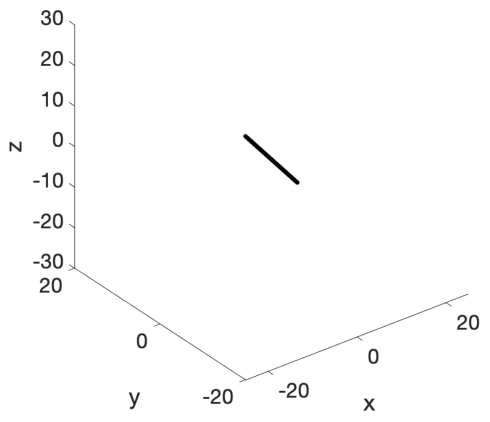
```
% Clean workspace
clear all; close all; clc

t = -5:0.05:5;
x = 5*t;
y = 4*t;
z = -6*t;
X = [x; y; z];

figure(4)
plot3(X(1,:),X(2,:),X(3,:),'k.','Markersize',10)
axis vis3d
hold on
xlabel('x')
ylabel('y')
zlabel('z')
set(gca,'Fontsize',16)
```

By construction, all of the data lies on a line. Let's look at what this means for the SVD.

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

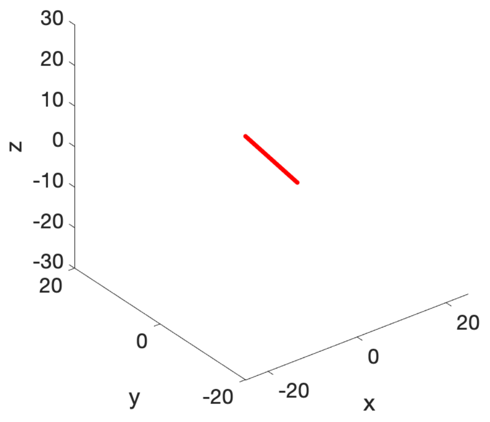


Our $S$ matrix comes out to be

$$S = \begin{bmatrix} 360.9221 & 0 & 0 \\ 0 & 0.0000 & 0 \\ 0 & 0 & 0.0000 \end{bmatrix}.$$

That is, there is only one nonzero singular value. From the theory presented in Lecture 11, this means that the rank of $X$ is 1. Equivalently, this means that the data falls on a line (which we already knew because we designed it that way). So, what does the rank-1 approximation look like?

```
X_rank1 = U(:,1)*S(1,1)*V(:,1)';
plot3(X_rank1(1,:),X_rank1(2,:),X_rank1(3,:),'r.','Markersize',10)
```



The rank-1 approximation falls exactly on the data points! I know you think this example is trivial - that's because it is. But, it is only trivial because we have three-dimensional data and we can plot it nicely. Imagine instead you had 100 different dimensions of data. Would it be obvious to you just by looking at the data matrix that the data is truly only 20 dimensional? With the SVD the answer is now yes!