Ch 5 Lecture 3

Intuition about SVD

Thinking again about matrix diagonalization

The same logic, applied to a general matrix

Many parts of this inspired by this blog post

Recap

$$A = USV^T$$

where

$$\underset{U}{\begin{pmatrix} | & & | \\ \mathbf{u}_1 & \cdots & \mathbf{u}_m \\ | & & | \end{pmatrix}} \quad \underset{V}{\begin{pmatrix} | & & | \\ \mathbf{v}_1 & \cdots & \mathbf{v}_n \\ | & & | \end{pmatrix}} \quad \underset{S}{\begin{pmatrix} \sigma_1 & & & & & \\ & \sigma_2 & & & & \\ & & \ddots & & & \\ & & & \sigma_r & & \\ & & & & \ddots & \\ & & & & & 0 \end{pmatrix}}$$

eigenvectors of AA^T A^TA

Example

$$A = \begin{pmatrix} 3 & 2 & 2 \\ 2 & 3 & -2 \end{pmatrix}$$

. . .

$$AA^T = \begin{pmatrix} 17 & 8 \\ 8 & 17 \end{pmatrix}, \quad A^TA = \begin{pmatrix} 13 & 12 & 2 \\ 12 & 13 & -2 \\ 2 & -2 & 8 \end{pmatrix}$$

$$AA^T = \begin{pmatrix} 17 & 8 \\ 8 & 17 \end{pmatrix} \qquad A^TA = \begin{pmatrix} 13 & 12 & 2 \\ 12 & 13 & -2 \\ 2 & -2 & 8 \end{pmatrix}$$

eigenvalues: $\lambda_1 = 25, \lambda_2 = 9$ eigenvalues: $\lambda_1 = 25, \lambda_2 = 9, \lambda_3 = 0$

eigenvectors eigenvectors

$u_1 = \binom{1/\sqrt{2}}{1/\sqrt{2}}$ $u_2 = \binom{1/\sqrt{2}}{-1/\sqrt{2}}$ $v_1 = \begin{pmatrix} 1/\sqrt{2} \\ 1/\sqrt{2} \\ 0 \end{pmatrix}$ $v_2 = \begin{pmatrix} 1/\sqrt{18} \\ -1/\sqrt{18} \\ 4/\sqrt{18} \end{pmatrix}$ $v_3 = \begin{pmatrix} 2/3 \\ -2/3 \\ -1/3 \end{pmatrix}$

SVD decomposition of A:

$$A = USV^T = \begin{pmatrix} 1/\sqrt{2} & 1/\sqrt{2} \\ 1/\sqrt{2} & -1/\sqrt{2} \end{pmatrix} \begin{pmatrix} 5 & 0 & 0 \\ 0 & 3 & 0 \end{pmatrix} \begin{pmatrix} 1/\sqrt{2} & 1/\sqrt{2} & 0 \\ 1/\sqrt{18} & -1/\sqrt{18} & 4/\sqrt{18} \\ 2/3 & -2/3 & -1/3 \end{pmatrix}$$

Reformulating SVD

SVD as a sum of rank-one matrices

$$A = \sigma_l u_l v_l^T + ... + \sigma_r u_r v_r^T$$

Back to our example

$$A = \begin{pmatrix} 3 & 2 & 2 \\ 2 & 3 & -2 \end{pmatrix}$$

. . .

Can decompose as

$$= 5 \begin{pmatrix} 1/\sqrt{2} \\ 1/\sqrt{2} \end{pmatrix} \begin{pmatrix} 1/\sqrt{2} & 1/\sqrt{2} & 0 \end{pmatrix} + 3 \begin{pmatrix} 1/\sqrt{2} \\ -1/\sqrt{2} \end{pmatrix} \begin{pmatrix} 1/\sqrt{18} & -1/\sqrt{18} & 4/\sqrt{18} \end{pmatrix}$$

. . .

$$A = \begin{pmatrix} 3 & 2 & 2 \\ 2 & 3 & -2 \end{pmatrix}$$

Uses of SVD

Moore-Penrose pseudoinverse

If we have a linear system

$$Ax = b$$

and A is invertible, then we can solve for x:

$$x = A^{-1}b$$

. . .

If A is not invertible, we can define instead a *pseudoinverse* A^+

. . .

Define A^+ in order to minimize the least squares error:

$$\|\mathbf{A}\mathbf{A}^+ - \mathbf{I_n}\|_2$$

. . .

Then we can estimate x as

$$\begin{aligned} Ax &= b \\ x &\approx A^+ b \end{aligned}$$

Finding the form of the pseudoinverse

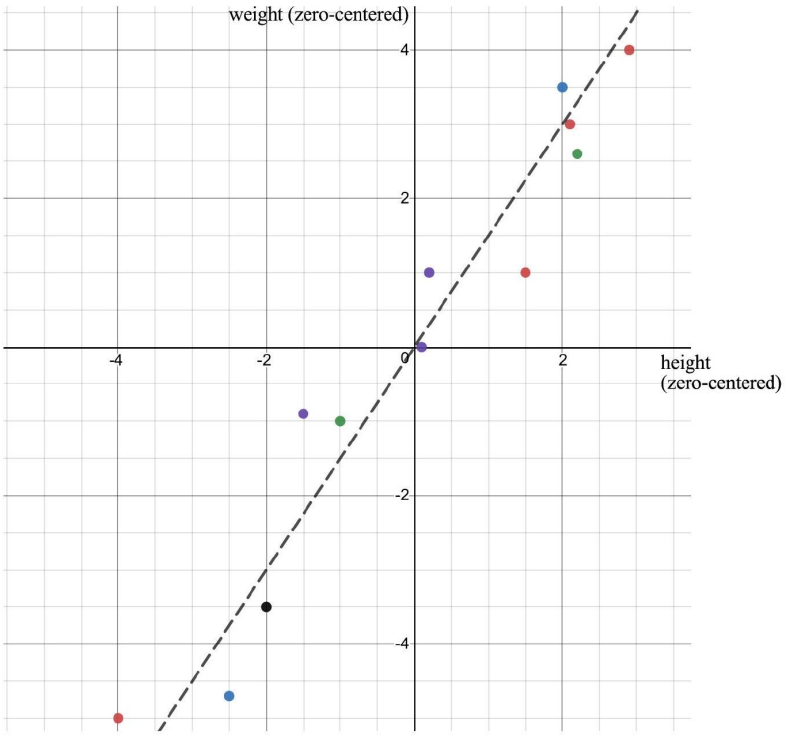
Example

Matrices as data

Example: Height and weight

$$A^T = \begin{bmatrix} 2.9 & -1.5 & 0.1 & -1.0 & 2.1 & -4.0 & -2.0 & 2.2 & 0.2 & 2.0 & 1.5 & -2.5 \\ 4.0 & -0.9 & 0.0 & -1.0 & 3.0 & -5.0 & -3.5 & 2.6 & 1.0 & 3.5 & 1.0 & -4.7 \end{bmatrix}$$

. . .



Covariance matrix

Covariance:

$$\sigma_{ab}^2 = \operatorname{cov}(a, b) = \mathrm{E}[(a - \bar{a})(b - \bar{b})]$$
$$\sigma_a^2 = \operatorname{var}(a) = \operatorname{cov}(a, a) = \mathrm{E}\left[(a - \bar{a})^2\right]$$

. . .

Covariance matrix:

$$= \begin{pmatrix} E\left[(x_1-\mu_1)(x_1-\mu_1)\right] & E\left[(x_1-\mu_1)(x_2-\mu_2)\right] & \dots & E\left[(x_1-\mu_1)(x_p-\mu_p)\right] \\ E\left[(x_2-\mu_2)(x_1-\mu_1)\right] & E\left[(x_2-\mu_2)(x_2-\mu_2)\right] & \dots & E\left[(x_2-\mu_2)(x_p-\mu_p)\right] \\ \vdots & \vdots & \ddots & \vdots \\ E\left[(x_p-\mu_p)(x_1-\mu_1)\right] & E\left[(x_p-\mu_p)(x_2-\mu_2)\right] & \dots & E\left[(x_p-\mu_p)(x_p-\mu_p)\right] \end{pmatrix}$$

. . .

$$\Sigma = \mathrm{E}\left[(X-\bar{X})(X-\bar{X})^{\mathrm{T}}\right]$$

. . .

$$\Sigma = \frac{XX^{\mathrm{T}}}{n} \quad \text{(if } X \text{ is already zero centered)}$$

. . .

For our dataset,

$$\text{Sample covariance } S^2 = \frac{A^T A}{(\mathrm{n}-1)} = \frac{1}{11}\begin{bmatrix} 53.46 & 73.42 \\ 73.42 & 107.16 \end{bmatrix}$$

$$A^T = \begin{bmatrix} 2.9 & -1.5 & 0.1 & -1.0 & 2.1 & -4.0 & -2.0 & 2.2 & 0.2 & 2.0 & 1.5 & -2.5 \\ 4.0 & -0.9 & 0.0 & -1.0 & 3.0 & -5.0 & -3.5 & 2.6 & 1.0 & 3.5 & 1.0 & -4.7 \end{bmatrix}$$

. . .

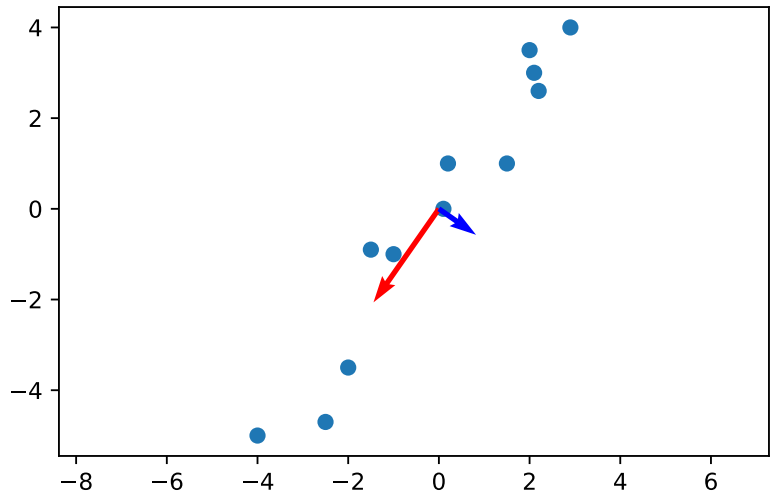
Plotting

The columns of the V matrix:

```
import numpy as np
import matplotlib.pyplot as plt
data = np.array([[2.9, 4.0], [-1.5, -0.9], [0.1, 0.0], [-1.0, -1.0], [2.1, 3.0], [-4.0, -5.0],
Ud,Sd,Vd=np.linalg.svd(data)
print(Vd)
plt.clf()

plt.scatter(data[:,0],data[:,1])
plt.quiver(0,0,Vd.T[0,0]*Sd[0]/5,Vd.T[1,0]*Sd[0]/5,angles='xy',scale_units='xy',scale=1, color
plt.quiver(0,0,Vd.T[0,1],Vd.T[1,1],angles='xy',scale_units='xy',scale=1, color = 'blue')
plt.axis('equal')
plt.show()
```

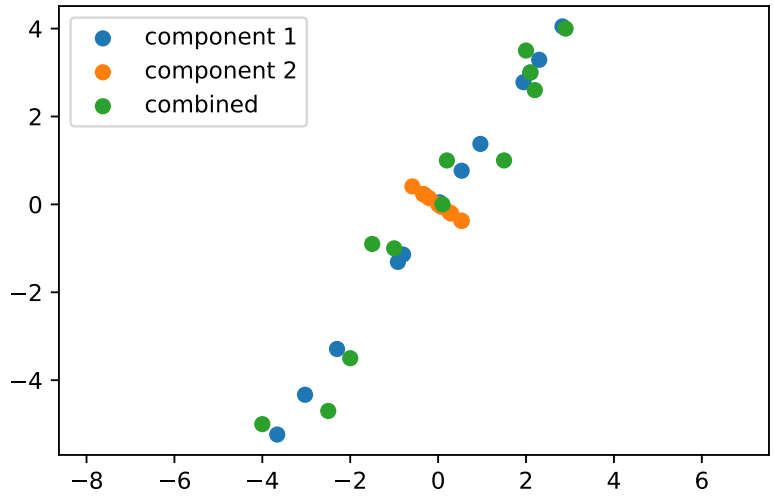
```
[[-0.57294952 -0.81959066]
 [ 0.81959066 -0.57294952]]
```



Walk through why these vectors are actually the eigenvectors of the covariance matrix.

The components of the data matrix

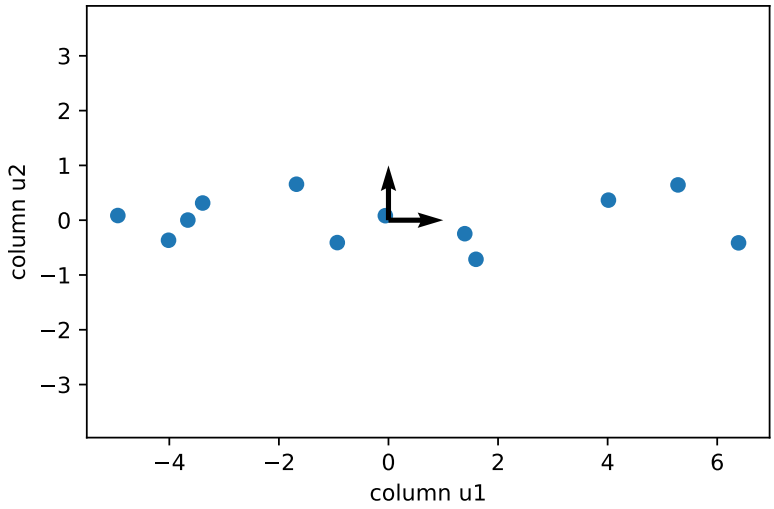
```
#plt.scatter(data[:,0],data[:,1])
plt.clf()
component1 = np.outer(Ud[:,0],Sd[0]*Vd[0])
component2 = np.outer(Ud[:,1],Sd[1]*Vd[1])
plt.scatter(component1[:,0],component1[:,1])
plt.scatter(component2[:,0],component2[:,1])
combined = component1 + component2
plt.scatter(combined[:,0],combined[:,1])
plt.axis('equal')
# add a legend
plt.legend(["component 1", "component 2", "combined"])
plt.show()
```



The columns of the U matrix

```
plt.scatter(Ud.T[0]*Sd[0],Ud.T[1]*Sd[1])
plt.quiver(0,0,0,1,angles='xy',scale_units='xy',scale=1)
plt.quiver(0,0,1,0,angles='xy',scale_units='xy',scale=1)
plt.axis('equal')
# give x label "column u1" and y label "column u2"
plt.xlabel("column u1")
plt.ylabel("column u2")
```

```
plt.show()
```

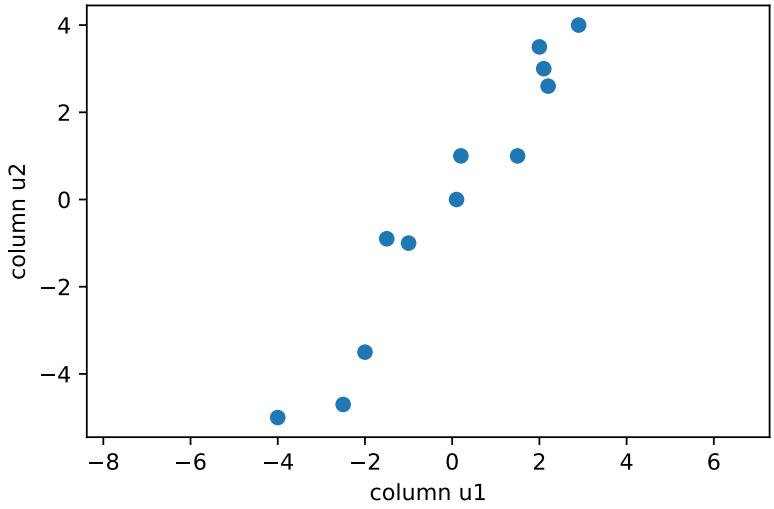


What if we had different orthogonal matrix for V, that wasn't eigenvectors of the covariance matrix?

```
va=np.array([[1,0],[0,1]])
ua=np.matmul(data,va)
plt.scatter(ua[:,0],ua[:,1])
print(np.cov(ua.T))
plt.axis('equal')
# give x label "column u1" and y label "column u2"
plt.xlabel("column u1")
plt.ylabel("column u2")

plt.show()
```

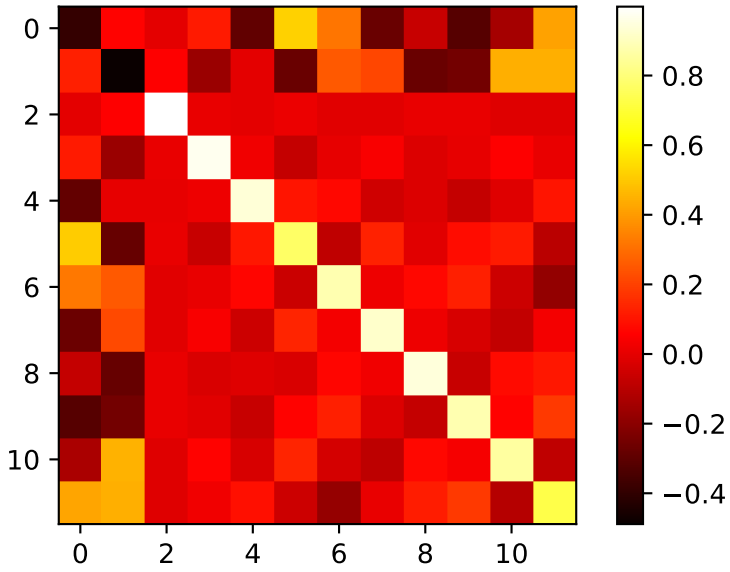
```
[[4.86       6.67454545]
 [6.67454545 9.74181818]]
```



The columns of the U matrix are no longer orthogonal.

The U matrix as a heat plot

```
# make a heatmap of the Ud matrix
plt.imshow(Ud, cmap='hot', interpolation='nearest')
plt.colorbar()
plt.show()
```


0
2
4
6
8
10
0
2
4
6
8
10
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4