Unsupervised Learning: Dimension Reduction

Industrial AI Lab.

Prof. Seungchul Lee

Dimension Reduction

- Motivation:
  - Can we describe high-dimensional data in a “simpler” way?

→ Dimension reduction without losing too much information

→ Find a low-dimensional, yet useful representation of the data

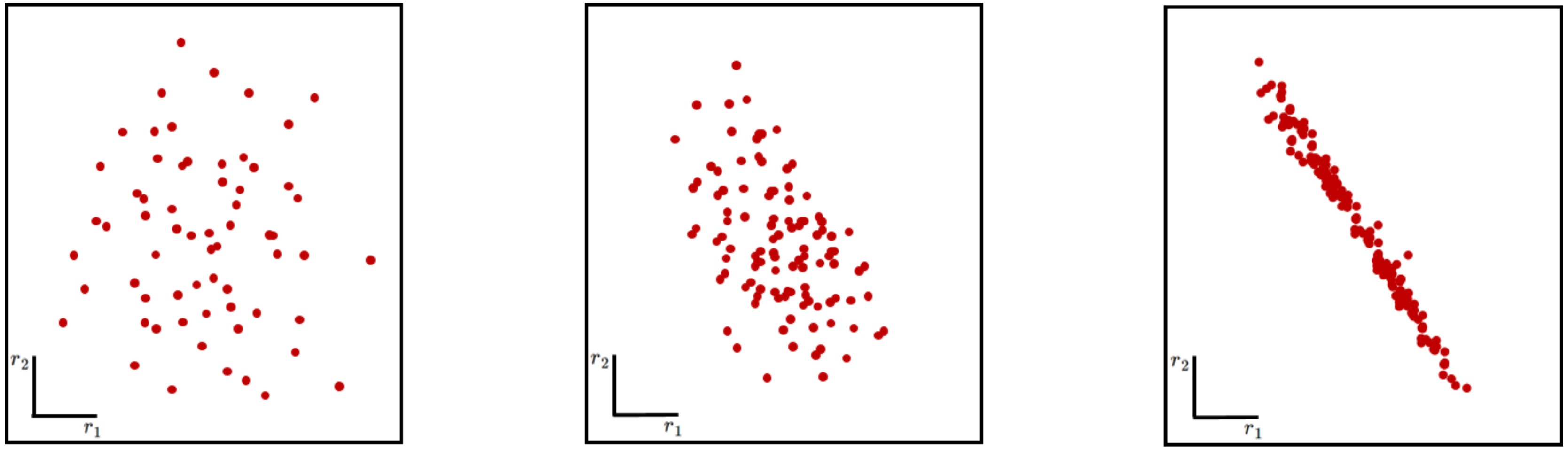
Dimension Reduction

- Why dimensionality reduction?
  - insights into the low-dimensional structures in the data (visualization)
  - Fewer dimensions $\Rightarrow$ Less chances of overfitting $\Rightarrow$ Better generalization
  - Speeding up learning algorithms
    - Most algorithms scale badly with increasing data dimensionality
  - Less storage requirements (data compression)

  - Note: Dimensionality reduction is different from feature selection
    - … although the goals are kind of the same
  - Dimensionality reduction is more like "feature extraction"
    - Constructing a small set of new features from the original features

Highly Correlated Data

- How?
  - idea: highly correlated data contains redundant features

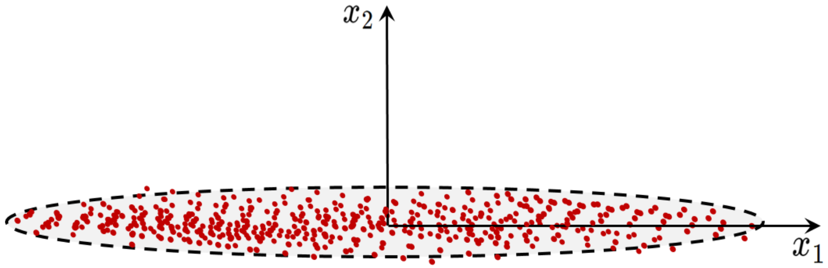


low redundancy

high redundancy

Principal Component Analysis (PCA)

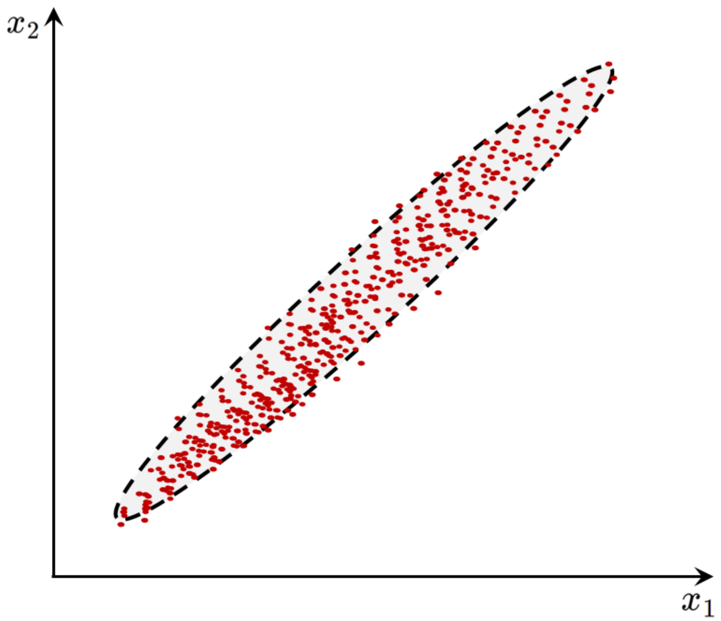
- Each example x has 2 features $\{x_1, x_2\}$
- Consider ignoring the feature x_2 for each example
- Each 2-dimensional example x now becomes 1-dimensional $x = \{x_1\}$
- Are we losing much information by throwing away x_2?



- No. Most of the data spread is along x_1 (very little variance along x_2)

Principal Component Analysis (PCA)

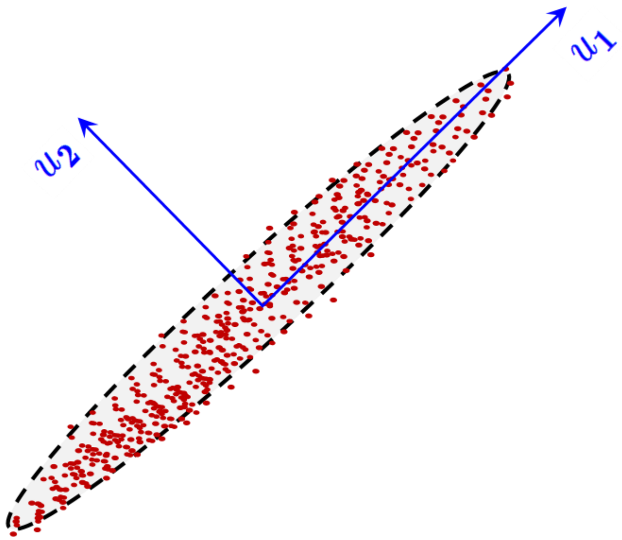
- Each example x has 2 features $\{x_1, x_2\}$
- Consider ignoring the feature x_2 for each example
- Each 2-dimensional example x now becomes 1-dimensional $x = \{x_1\}$
- Are we losing much information by throwing away x_2?



- Yes, the data has substantial variance along both features (i.e., both axes)

Principal Component Analysis (PCA)

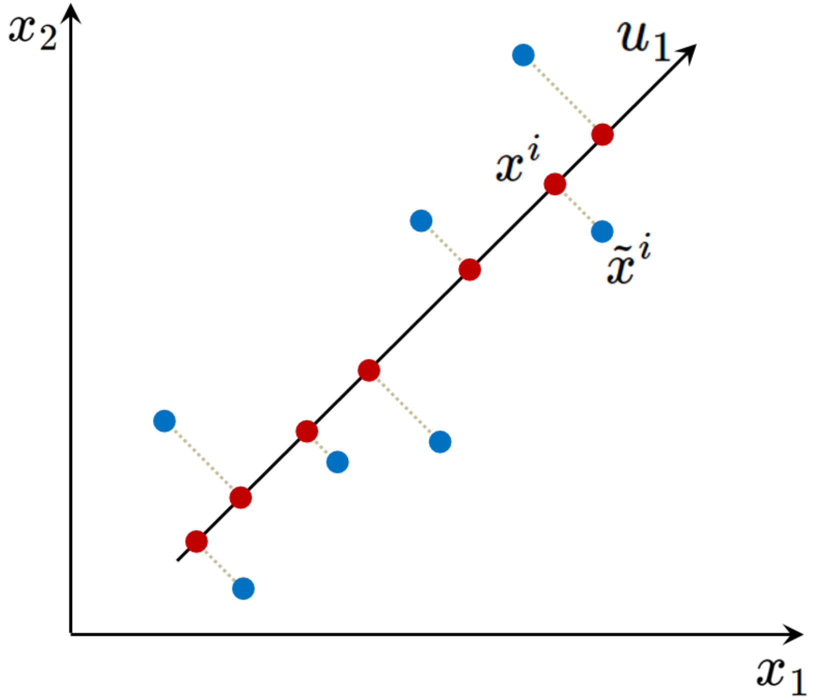
- Now consider a change of axes
- Each example x has 2 features $\{u_1, u_2\}$
- Consider ignoring the feature u_2 for each example
- Each 2-dimensional example x now become 1-dimensional $x = \{u_1\}$



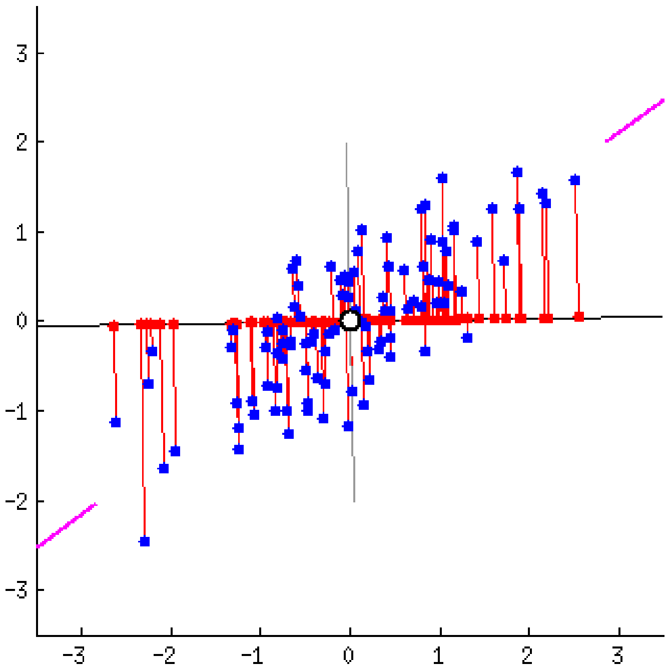
- No. Most of the data spread is along u_1 (very little variance along u_2)

Principal Component Analysis (PCA)

- Data → projection onto unit vector $\hat{u}_1$
  - PCA is used when we want projections capturing maximum variance directions
  - Principal Components (PC): directions u_1 of maximum variability in the data
  - Roughly speaking, PCA does a change of axes that can represent the data in a succinct manner



Principal Component Analysis (PCA)



- HOW?
  1. Maximize variance (most separable)
  2. Minimize the sum-of-squares (minimum squared error)

PCA Algorithm: Pre-processing

- Given data

$$x^{(i)} = \begin{bmatrix} x_1^{(i)} \\ \vdots \\ x_n^{(i)} \end{bmatrix}, \qquad X = \begin{bmatrix} \cdots & (x^{(1)})^T & \cdots \\ \cdots & (x^{(2)})^T & \cdots \\ & \vdots & \\ \cdots & (x^{(m)})^T & \cdots \end{bmatrix}$$

- Shifting (zero mean) and rescaling (unit variance)

1. Shift to zero mean

$$\mu = \frac{1}{m} \sum_{i=1}^{m} x^{(i)}$$

$$x^{(i)} \leftarrow x^{(i)} - \mu \quad \text{(zero mean)}$$

2. [optional] Rescaling (unit variance)

$$\sigma_j^2 = \frac{1}{m-1} \sum_{i=1} m \left(x_j^{(i)} \right)^2$$

$$x_j^{(i)} \leftarrow \frac{x_j^{(i)}}{\sigma_j}$$

PCA Algorithm: Maximize Variance

- Find unit vector u such that maximizes variance of projections
  - Note: $m \approx m-1$ for big data

$$
\begin{aligned}
\text{variance of projected data} &= \frac{1}{m}\sum_{i=1}^{m}\left(u^T x^{(i)}\right)^2 = \frac{1}{m}\sum_{i=1}^{m}\left({x^{(i)}}^T u\right)^2 \\
&= \frac{1}{m}\sum_{i=1}^{m}\left({x^{(i)}}^T u\right)^T\left({x^{(i)}}^T u\right) = \frac{1}{m}\sum_{i=1}^{m} u^T x^{(i)} {x^{(i)}}^T u \\
&= u^T\left(\frac{1}{m}\sum_{i=1}^{m} x^{(i)} {x^{(i)}}^T\right) u \\
&= u^T S u \qquad \left(S = \frac{1}{m} X^T X : \text{sample covariance matrix}\right)
\end{aligned}
$$

Maximize Variance

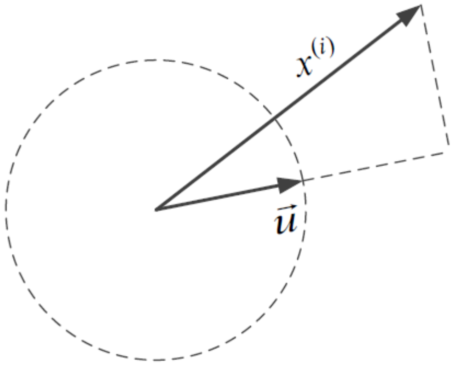
- In an optimization form

$$\begin{aligned} \text{maximize} \quad & u^T S u \\ \text{subject to} \quad & u^T u = 1 \end{aligned}$$

$$u^T S u = u^T \lambda u = \lambda u^T u = \lambda \quad (\text{Eigen analysis}: Su = \lambda u)$$

$$\begin{aligned} &\Longrightarrow \text{ pick the largest eigenvalue } \lambda_1 \text{ of covariance matrix } S \\ &\Longrightarrow \; u = u_1 \text{ is the } \lambda_1's \text{ corresponding eigenvector} \\ &\Longrightarrow \; u_1 \text{ is the first principal component (direction of highest variance in the data)} \end{aligned}$$

Minimize the Sum-of-Squared Error



$$
\begin{aligned}
\|e^{(i)}\|^2 &= \|x^{(i)}\|^2 - \left({x^{(i)}}^T u\right)^2 \\
&= \|x^{(i)}\|^2 - \left({x^{(i)}}^T u\right)^T \left({x^{(i)}}^T u\right) \\
&= \|x^{(i)}\|^2 - u^T x^{(i)} {x^{(i)}}^T u
\end{aligned}
$$

$$
\begin{aligned}
\frac{1}{m}\sum_{i=1}^{m}\|e^{(i)}\|^2 &= \frac{1}{m}\sum_{i=1}^{m}\|x^{(i)}\|^2 - \frac{1}{m}\sum_{i=1}^{m} u^T x^{(i)} {x^{(i)}}^T u \\
&= \frac{1}{m}\sum_{i=1}^{m}\|x^{(i)}\|^2 - u^T \left(\frac{1}{m}\sum_{i=1}^{m} x^{(i)} {x^{(i)}}^T\right) u
\end{aligned}
$$

Minimize the Sum-of-Squared Error

- In an optimization form

$$\text{minimize } \underbrace{\frac{1}{m}\sum_{i=1}^{m}\|x^{(i)}\|^2}_{\text{constant given } x_i} - \underbrace{u^T\left(\frac{1}{m}\sum_{i=1}^{m}x^{(i)}x^{(i)^T}\right)u}_{\text{maximize}}$$

$$\Longrightarrow \text{maximize } u^T\left(\frac{1}{m}\sum_{i=1}^{m}x^{(i)}x^{(i)^T}\right)u = \max \; u^T S u$$

$$\therefore \text{ minimize } error^2 = \text{maximize } variance$$

Dimension Reduction Method ($n \to k$)

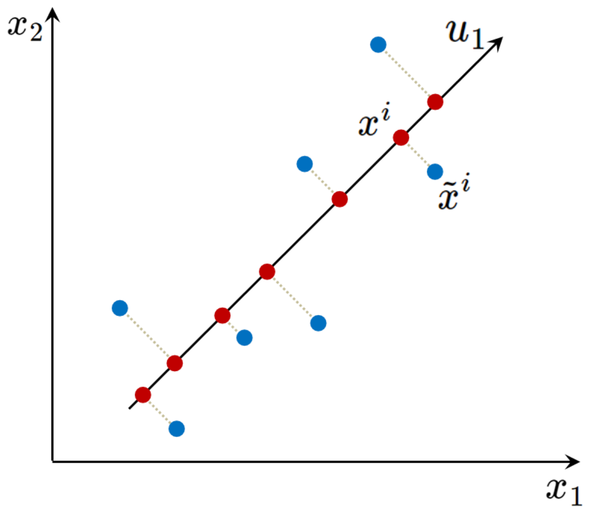
1. Choose top k (orthonormal) eigenvectors, $U = [u_1, u_2, \cdots, u_k]$
2. Project x_i onto span $\{u_1, u_2, \cdots, u_k\}$

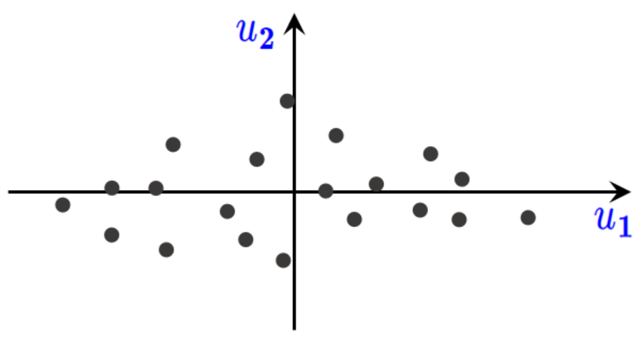
$$z^{(i)} = \begin{bmatrix} u_1^T x^{(i)} \\ u_2^T x^{(i)} \\ \vdots \\ u_k^T x^{(i)} \end{bmatrix} \quad \text{or} \quad z = U^T x$$

- $x^{(i)}$ → projection onto unit vector $u \Longrightarrow u^T x^{(i)}$ = distance from the origin along u

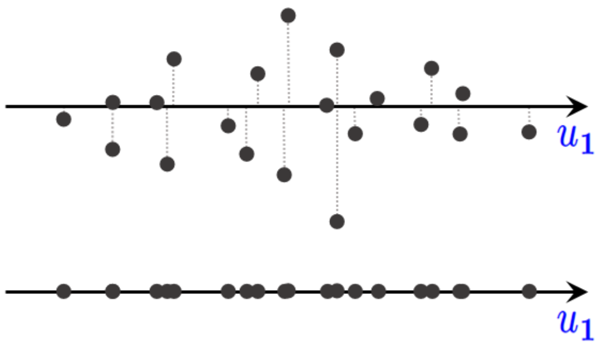
Principal Component Analysis

- Data → projection onto unit vector u



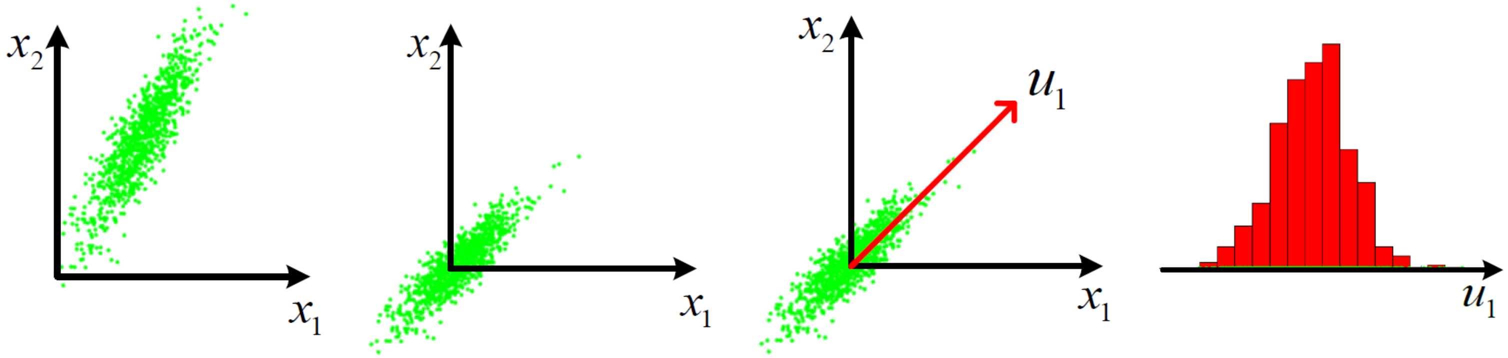


dimension 2개 선택할 경우



dimension 1개만 선택할 경우

Pictorial Summary of PCA



Pre-processing

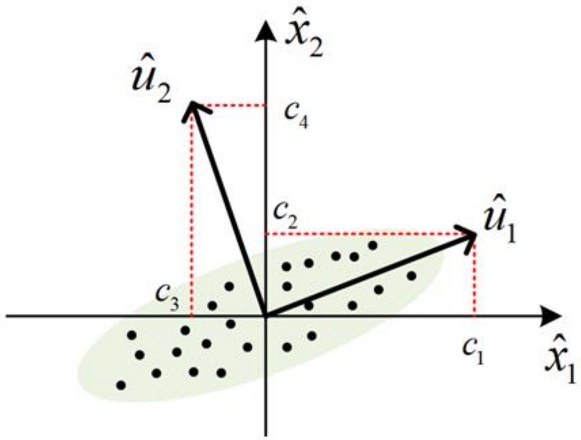
Eigenvector u

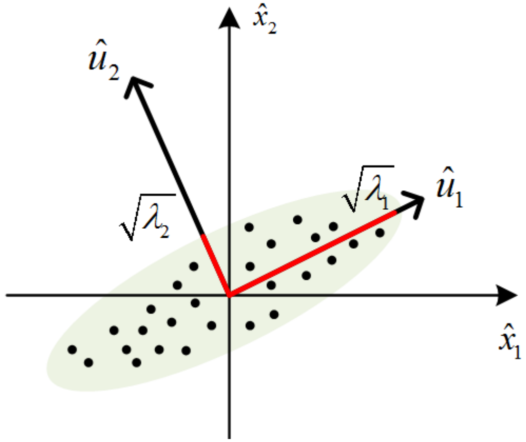
Projected (2D→1D)

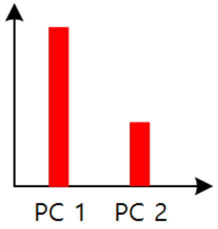
PCA Visualization

- Eigenvectors
  - Given basis $\{\hat{x}_1, \hat{x}_2\}$ to transformed basis $\{\hat{u}_1, \hat{u}_2\}$

$$[\hat{u}_1 \ \hat{u}_2] = [\hat{x}_1 \ \hat{x}_2] \begin{bmatrix} c_1 & c_3 \\ c_2 & c_4 \end{bmatrix}$$



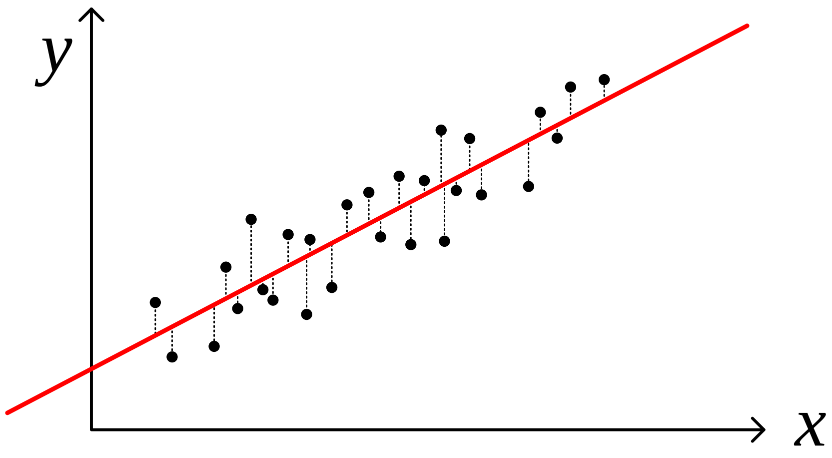




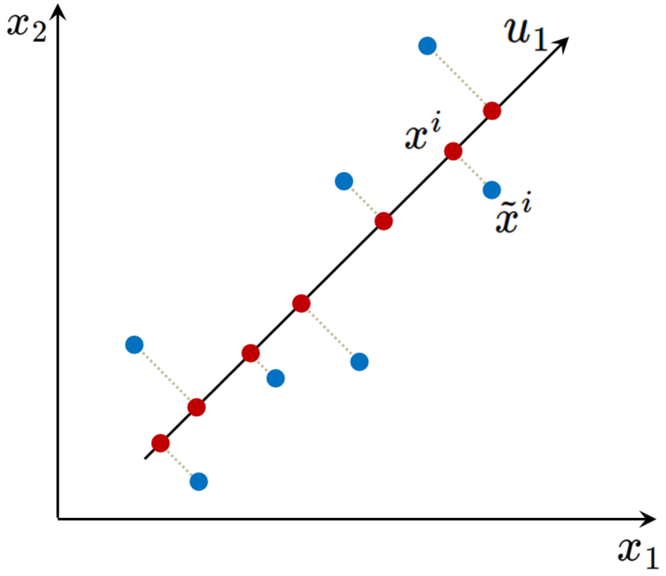
- Eigenvalues
  - λ_1, λ_2 indicates variance along the eigenvectors, respectively.
  - The larger eigenvalue is, the more dominant feature (eigenvector) is.

Linear Regression vs. PCA

Linear Regression

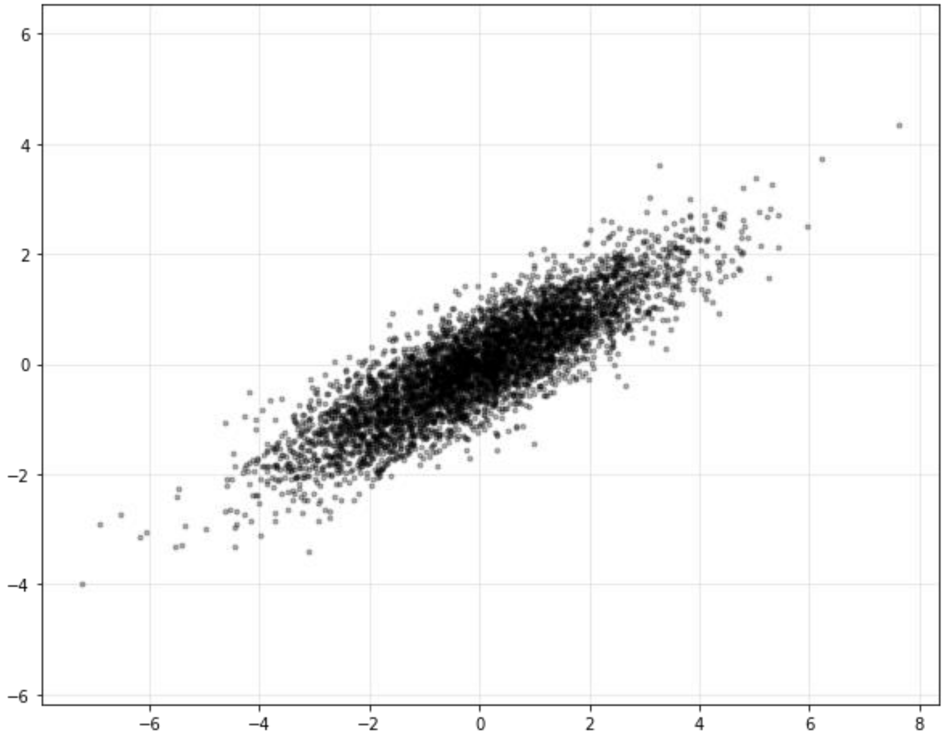


PCA



Python Codes

```
# data generation
m = 5000
mu = np.array([0, 0])
sigma = np.array([[3, 1.5],
                  [1.5, 1]])

X = np.random.multivariate_normal(mu, sigma, m)
X = np.asmatrix(X)
```



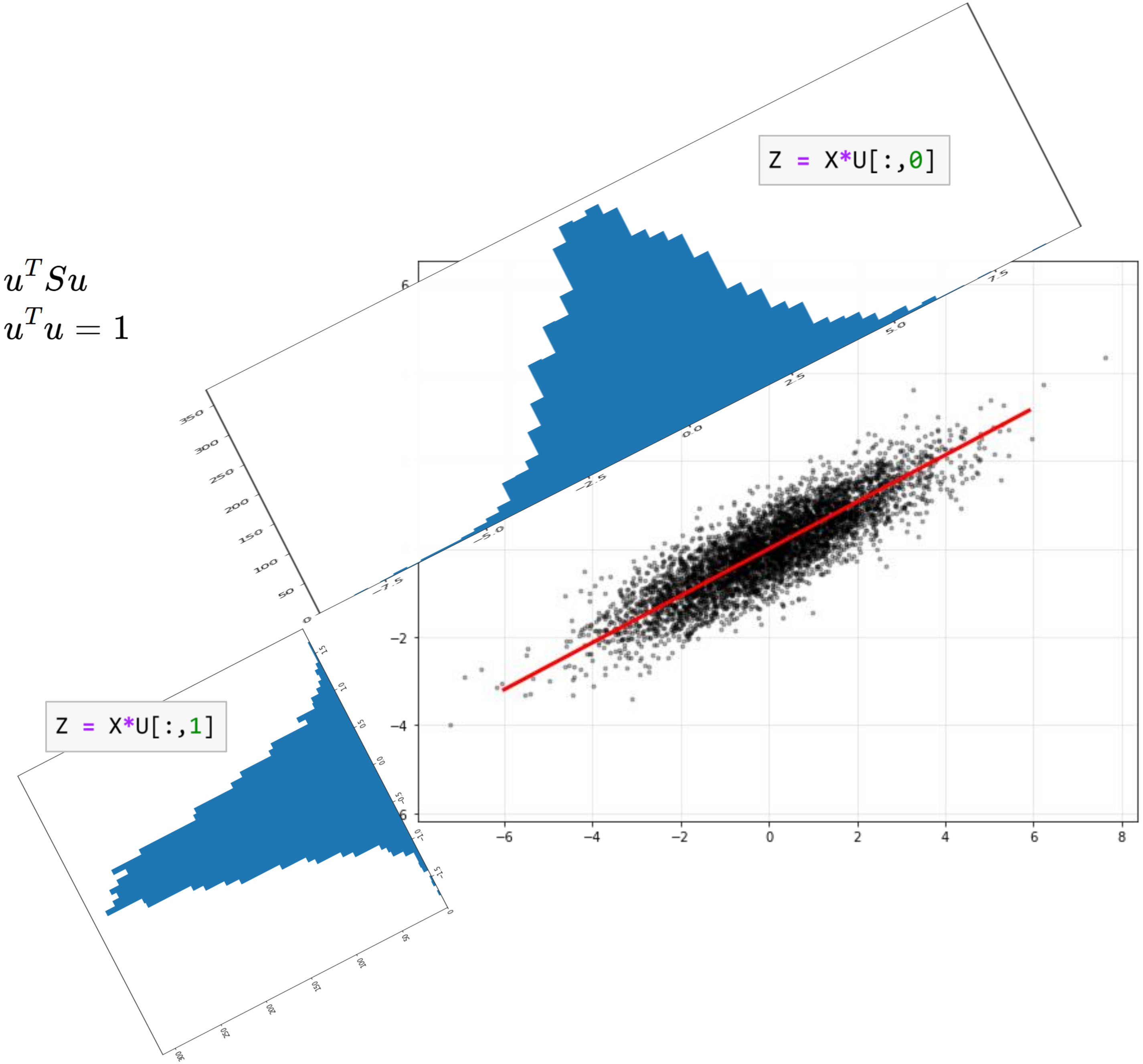
Python Codes

```
S = 1/(m-1)*X.T*X
S = np.asmatrix(S)

D, U = np.linalg.eig(S)

idx = np.argsort(-D)
D = D[idx]
U = U[:,idx]

print(D, '\n')
print(U)
```

```
[3.90595228 0.19884608]

[[ 0.88112544 -0.4728826 ]
 [ 0.4728826   0.88112544]]
```

$$\begin{aligned} \text{maximize} \quad & u^T S u \\ \text{subject to} \quad & u^T u = 1 \end{aligned}$$

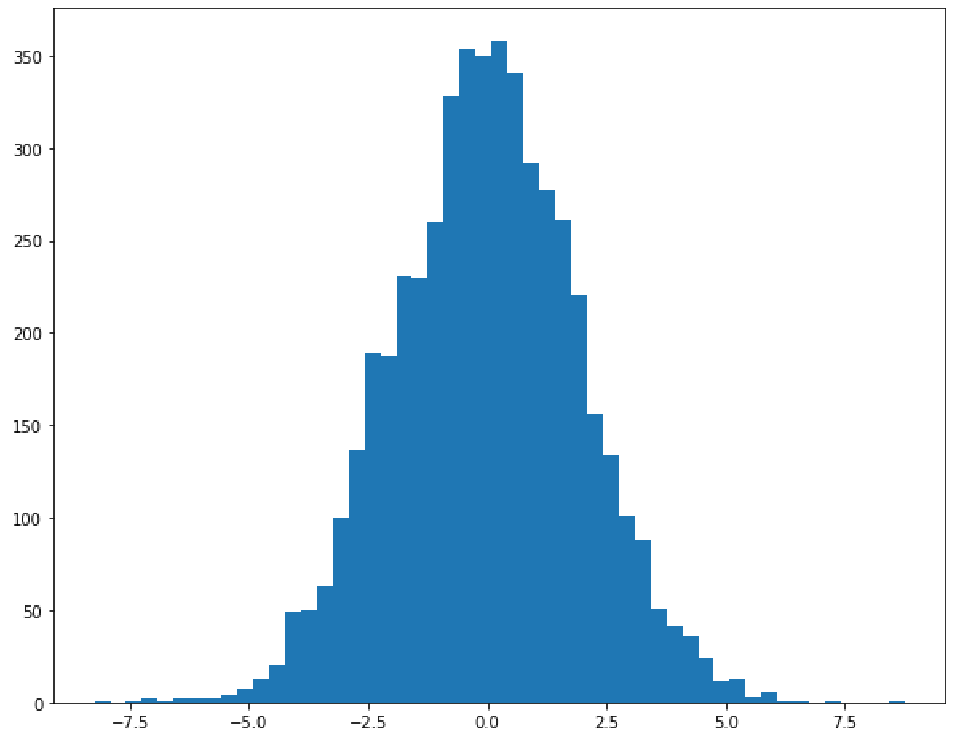
```
h = U[1,0]/U[0,0]
xp = np.arange(-6, 6, 0.1)
yp = h*xp
```



Scikit-learn

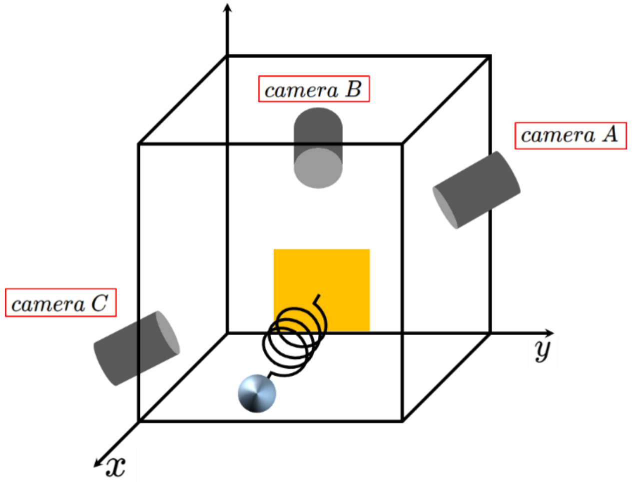
```python
from sklearn.decomposition import PCA

pca = PCA(n_components = 1)
pca.fit(X)
```

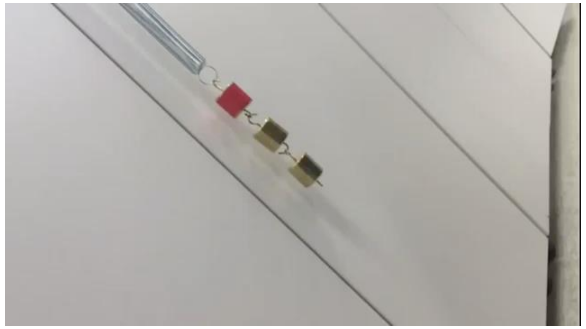
```python
u = pca.transform(X)

plt.figure(figsize = (10, 8))
plt.hist(u, 51)
plt.show()
```

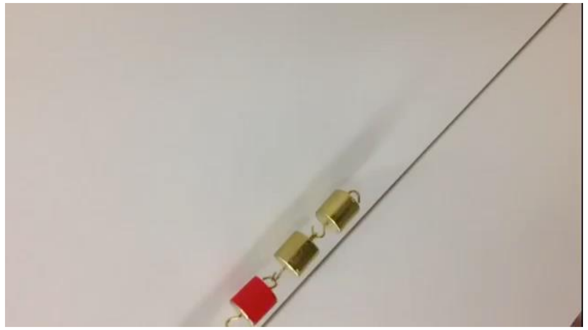


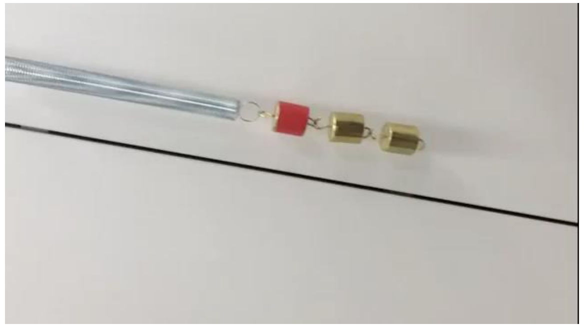
PCA Example

- Multiple video camera records of spring and mass system
- Optimal data representation
  - Find the most informative point of view





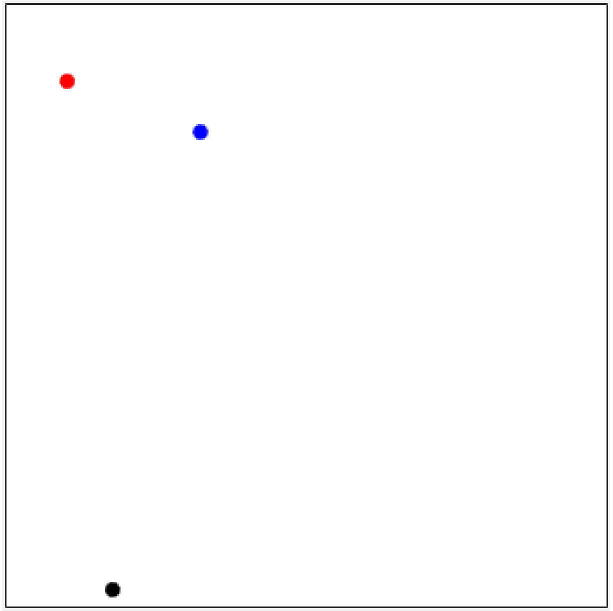




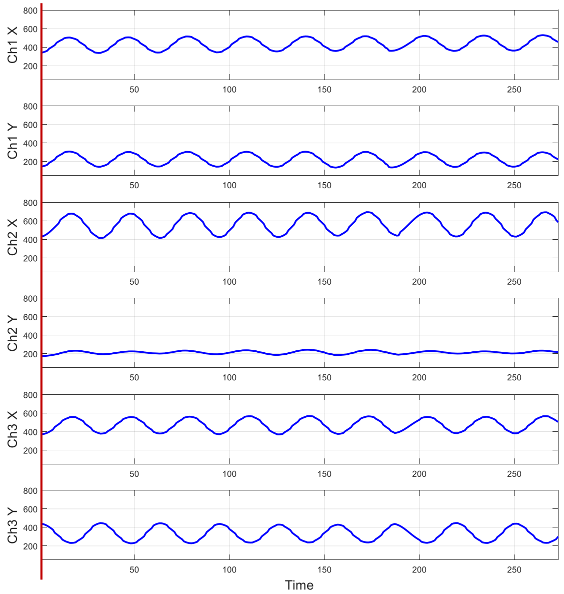
- source:
  - https://www.cs.princeton.edu/picasso/mats/PCA-Tutorial-Intuition_jp.pdf

Multivariate Time Series

- System order can be inferred from
  - Laws of physics or
  - Data



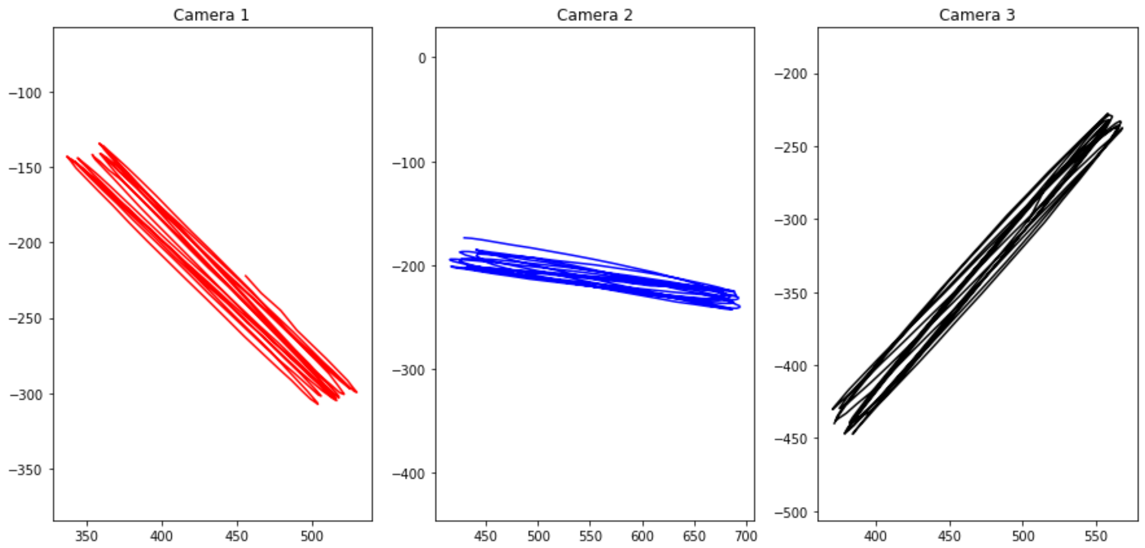
Measured observations



Multivariate Time Series

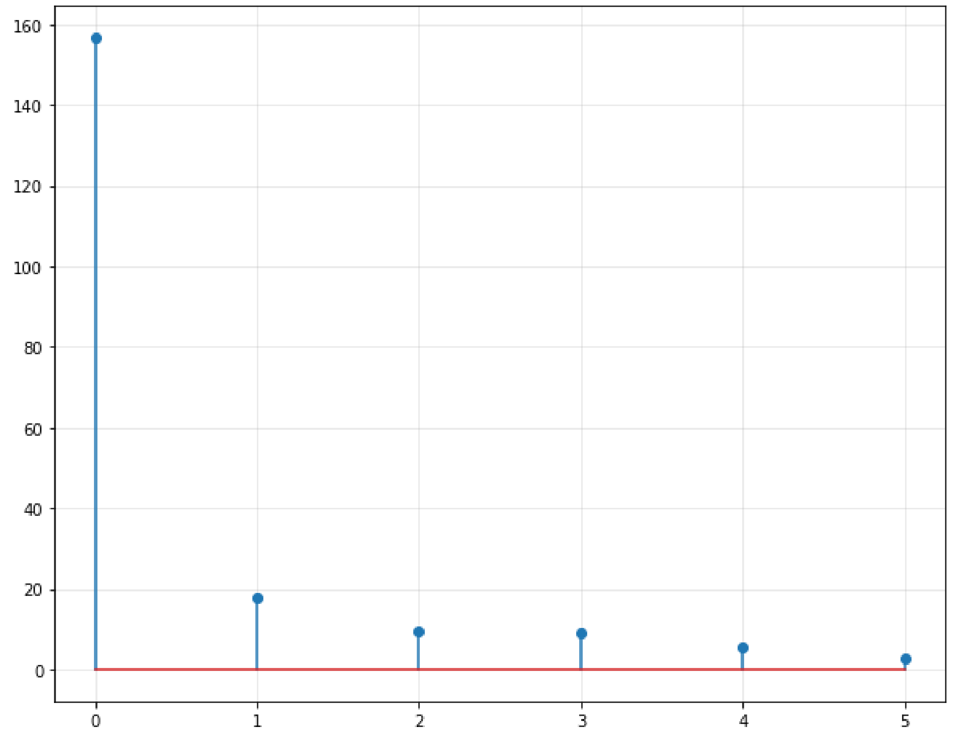
$$x^{(i)} = \begin{bmatrix} x \text{ in camera } 1 \\ y \text{ in camera } 1 \\ x \text{ in camera } 2 \\ y \text{ in camera } 2 \\ x \text{ in camera } 3 \\ y \text{ in camera } 3 \end{bmatrix}, \qquad X_{m\times 6} = \begin{bmatrix} \cdots & (x^{(1)})^T & \cdots \\ \cdots & (x^{(2)})^T & \cdots \\ & \vdots & \\ & \vdots & \\ \cdots & (x^{(m)})^T & \cdots \end{bmatrix}$$

$$\begin{aligned} \text{maximize} \quad & u^T S u \\ \text{subject to} \quad & u^T u = 1 \end{aligned}$$



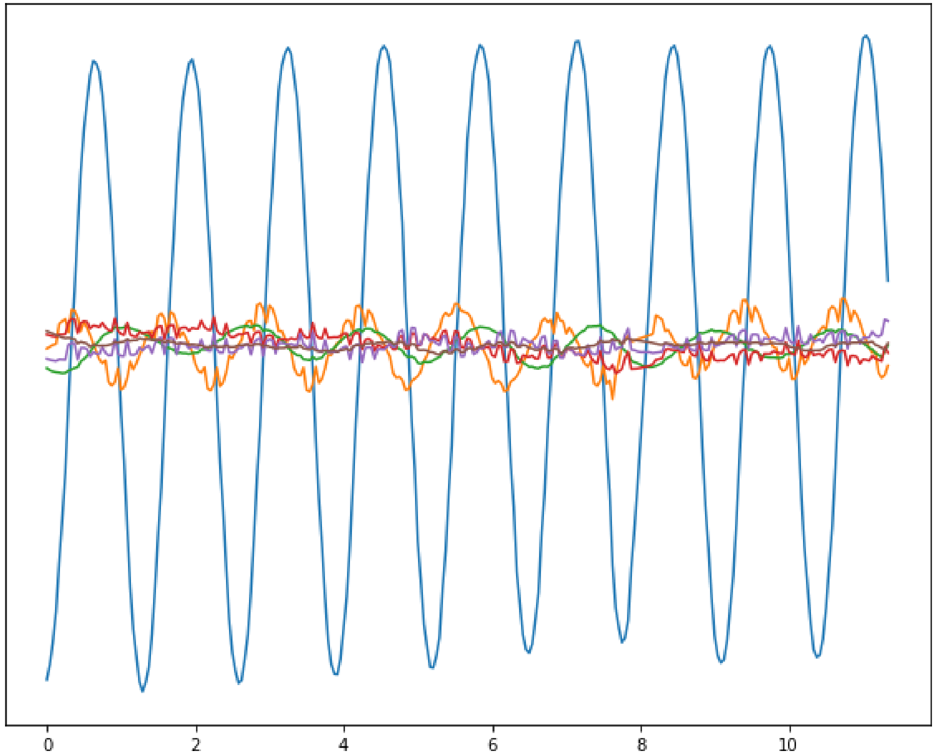
Eigenvalues

```
plt.figure(figsize = (10,8))
plt.stem(np.sqrt(D))
plt.grid(alpha = 0.3)
plt.show()
```

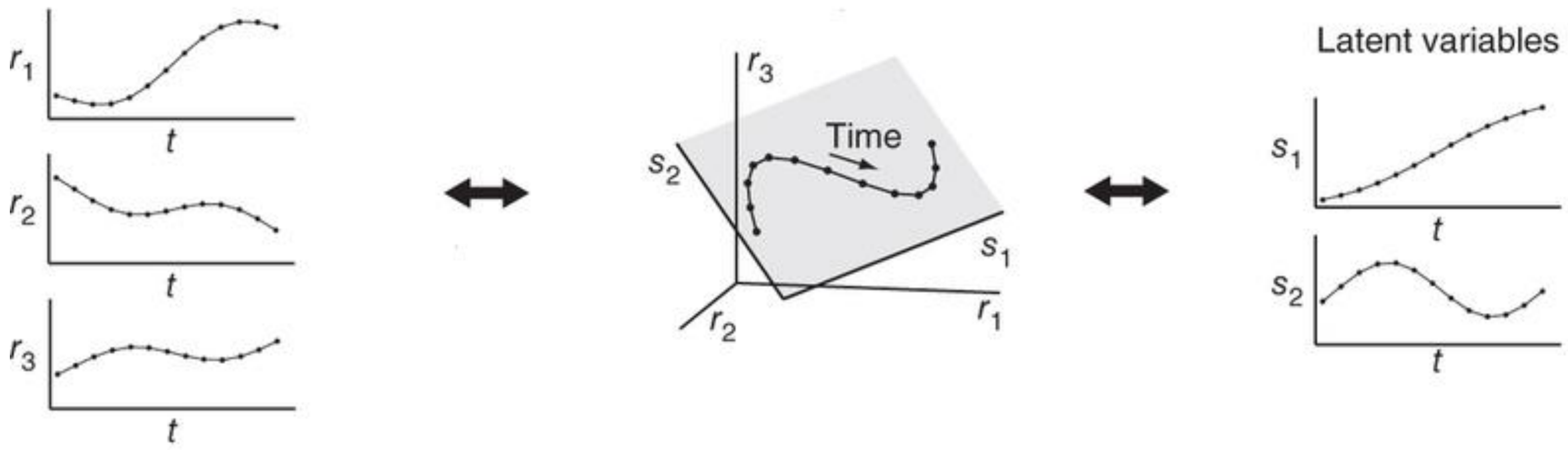


Projection onto Principal Components

```
# relative magnitutes of the principal components

Z = X*U
xp = np.arange(0, m)/24    # 24 frame rate
```



PCA Example



John P Cunningham & Byron M Yu, Dimensionality reduction for large-scale neural recordings, Nature Neuroscience 17, 1500–1509 (2014)