Feature selection and dimensional reduction

Feature selection

How would we know which feature to be selected, combined, or removed?

Feature selection

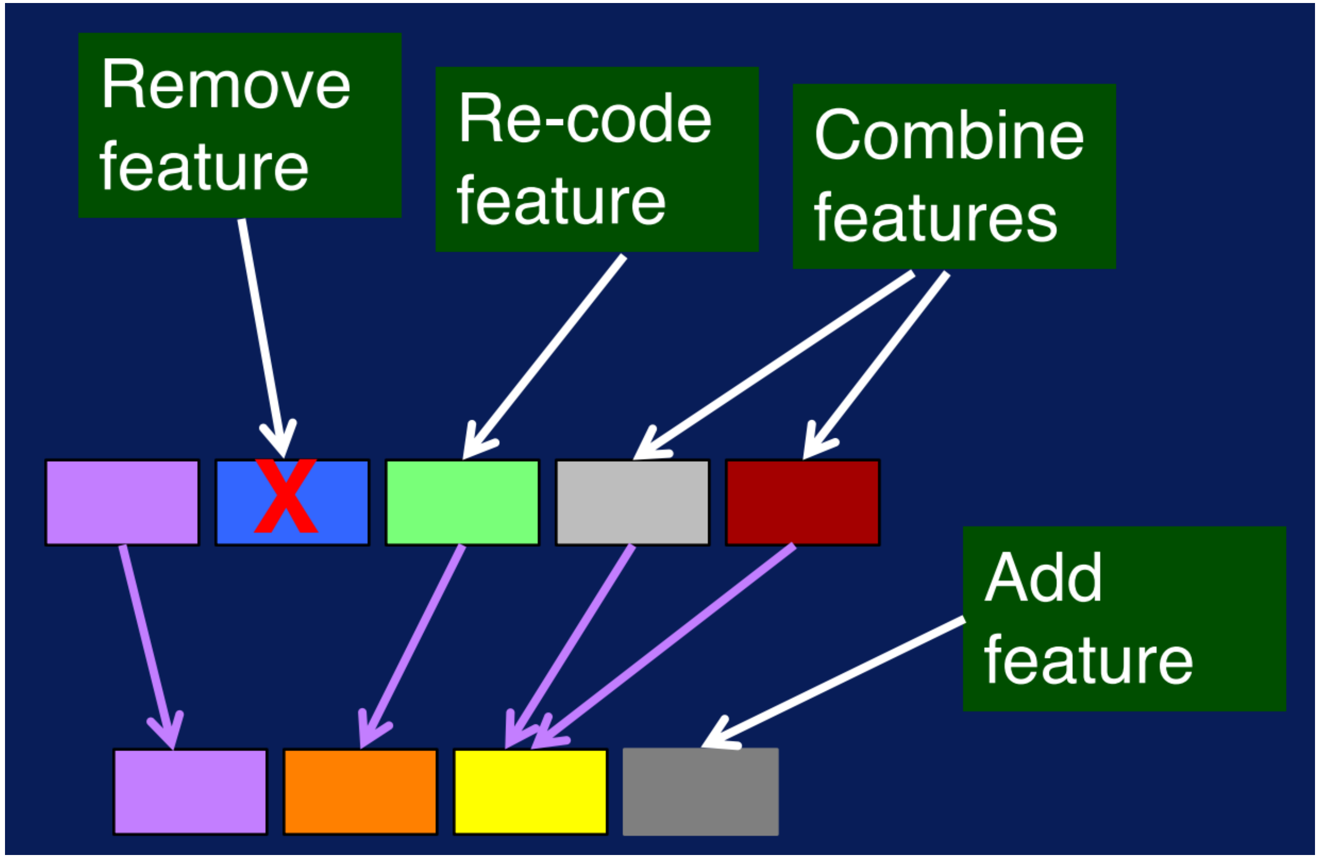
Large Dataset

- Huge amount of features are collected

- Not all are important

Ex: Removing / Filtering

Note: Removing carefully.



When should data be removed?

Feature selection

Removing features

- feature with std = 0 or variance = 0

ex. Remove 'Age' if

All students in the dataset

are in the same age / same year

-

When should data be combined?

Feature selection

Property type
Single family home
Townhouse
Condo

D1	D2
1	0
0	1
0	0

- D1: Is it a single family home?
- D2: Is it a townhouse?
- D1 = D2 = 0 indicates a condo.

Property type
Single family home
Townhouse
Condo
Coop
Multi-family
Mobile home

Property type
Single family home
Townhouse
Others

- Is it necessary to keep all categories?

UC Sandiego: https://www.coursera.org/learn/big-data-machine-learning/home/welcome

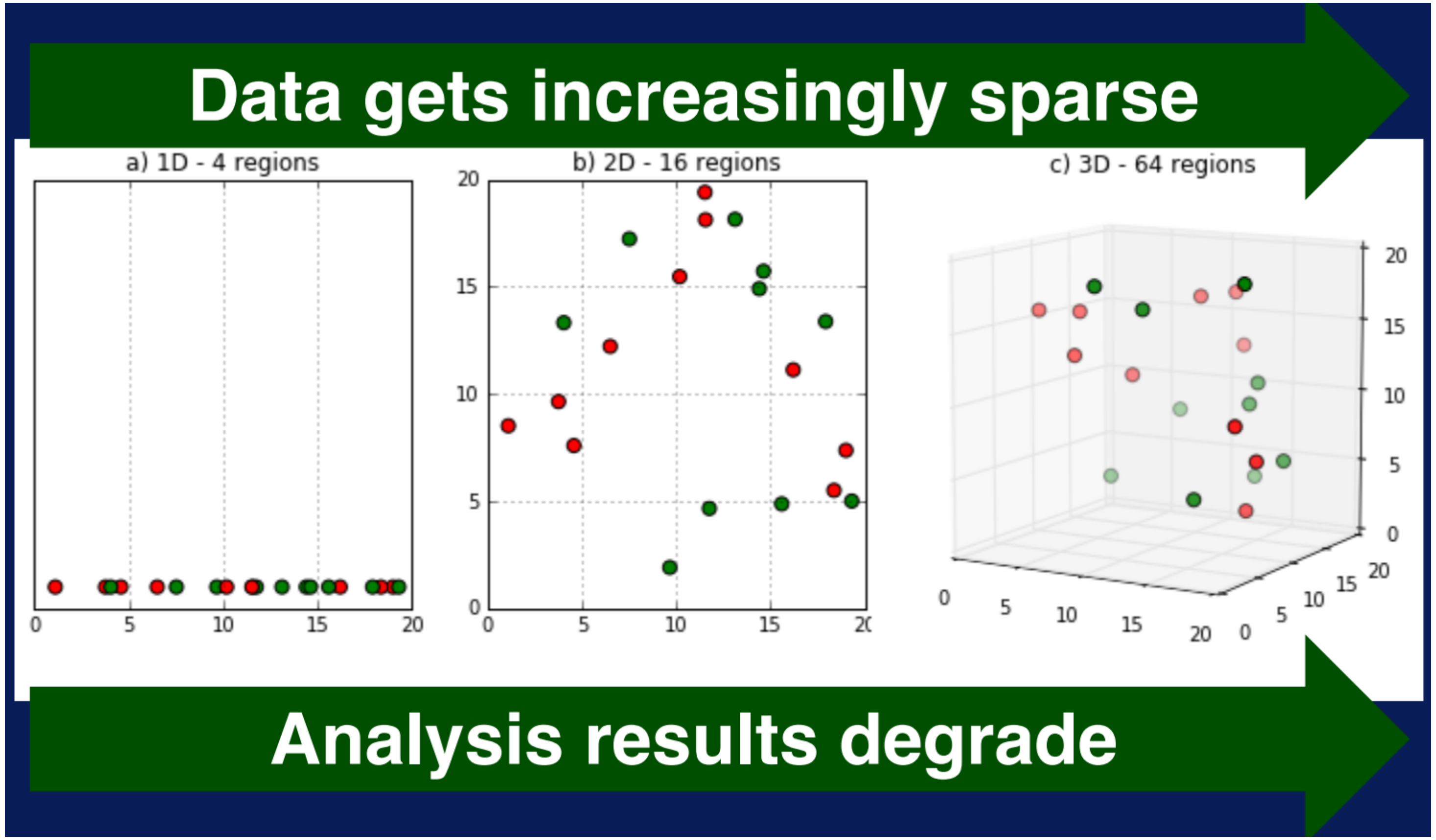
When should data be transformed or replaced?

Dimension reduction

How would we visualize the data?

Is the data perfect or noisy?

Dimension of data



- certain calculations used in analysis become much more difficult to define and calculate effectively

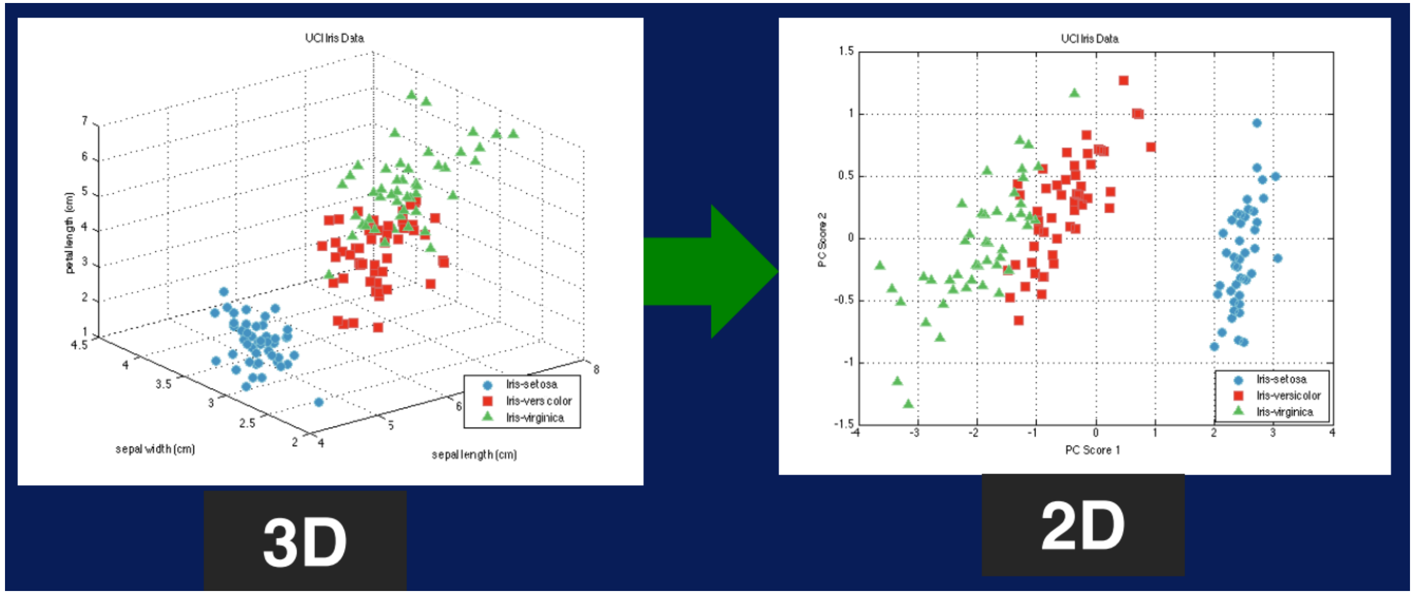
- distances between samples are harder to compare since all samples are far away from each other

- the difficulty of dealing with high dimensional data and as referred to as the curse of dimensionality

the number of dimensions increases,
the number of regions increases exponentially and
the data becomes increasingly sparse

UC Sandiego: https://www.coursera.org/learn/big-data-machine-learning/home/welcome

Dimensional reduction



Dimensional Reduction Techniques

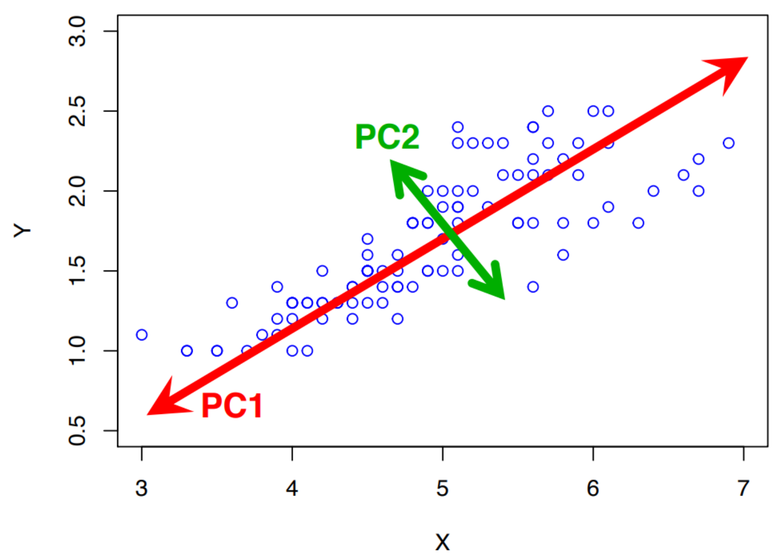
- Principle Component Analysis (PCA)

- Finding new mapping on to principle (significant) space (domain)

- Eigen-based Technique

UC Sandiego: https://www.coursera.org/learn/big-data-machine-learning/home/welcome

Dimensional reduction



Principle Component Analysis (PCA)

- Old representation

X-Y Coordinate

feature 1 – 2 Coordinate

- New PCA Coordinate

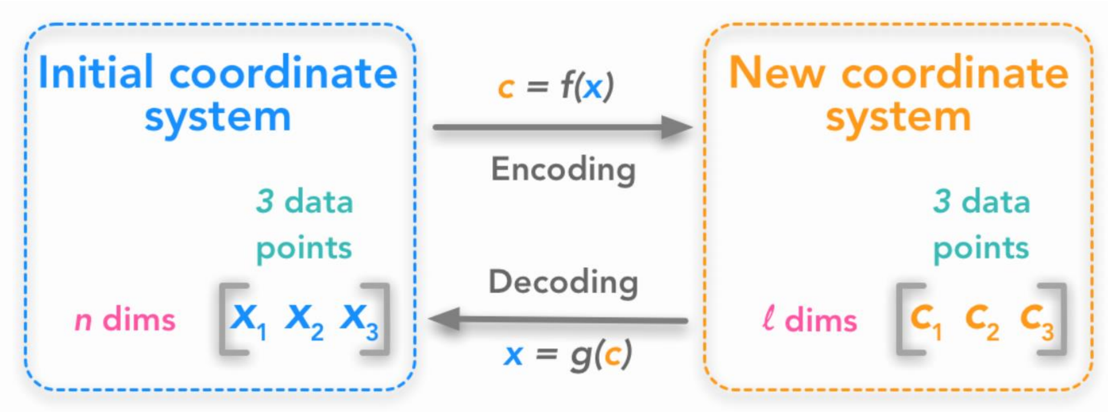
PC1 – PC2 Coordinate

PC1 = Eigen_Vector with max

lue

PC2 = Eigen_Vector with 2nd

lue



weighted sum of old coordinate

- difficult to interpret

https://hadrienj.github.io/posts/Deep-Learning-Book-Series-2.12-Example-Principal-Components-Analysis/

PCA:

Dimensional reduction

Feature Coordinate

Standardized data

- Zero mean
- Unit variance
- Covariance Matrix

PCA new coordinate

SVD (Singular Value Decomposition)

Eigen values

Eigen vectors

Organize PCA dominant dimension

Sorted

Eigen values

Eigen vectors

Selected dominant eigen

Selected

k-Eigen values

Corresponding k-Eigen vectors

PCA Transform

Dot_product (Input, k-Eigenvectors)

PCA.fit()

http://www.iro.umontreal.ca/~pift6080/H09/documents/papers/pca_tutorial.pdf

PCA:
#1: Standardized data

Feature Coordinate

Standardized data

- Zero mean
- Unit variance
- Covariance Matrix

transformation of the data onto

- mean subtraction

mean = 0

- unit scale

mean=0 and variance=1

	a	b
1	9	39
2	15	56
3	25	93
4	14	61
5	10	50
6	18	75
7	0	32
8	16	85
9	5	42
10	19	70

http://www.iro.umontreal.ca/~pift6080/H09/documents/papers/pca_tutorial.pdf

PCA:
#1: Standardized data

Feature Coordinate

Standardized data

- Zero mean
  - Unit variance
- Covariance Matrix

transformation of the data onto

- mean subtraction

mean = 0

- unit scale

mean=0 and variance=1

	H	M	Hn	Mn
1	9	39		
2	15	56		
3	25	93		
4	14	61		
5	10	50		
6	18	75		
7	0	32		
8	16	85		
9	5	42		
10	19	70		
sum	131	603		
u	13.1	60.3		
std	7.279	20.29		

http://www.iro.umontreal.ca/~pift6080/H09/documents/papers/pca_tutorial.pdf

PCA:

#2: Covariance matrix

Feature Coordinate

Standardized data

- Zero mean
  - Unit variance
- Covariance Matrix

	H	M	Hn	Mn	Hn.Hn	Mn.Mn	Hn.Mn
1	9	39					
2	15	56					
3	25	93					
4	14	61					
5	10	50					
6	18	75					
7	0	32					
8	16	85					
9	5	42					
10	19	70					
sum	131	603					
u	13.1	60.3					
std	7.279	20.29					

$$C = \begin{bmatrix} cov(HH) & cov(HM) \\ cov(MH) & cov(MM) \end{bmatrix}$$

$$cov(HH) = \frac{1}{N}\sum_{i=1}^{N}(H_i - \bar{H})(H_i - \bar{H})$$

$$cov(HM) = \frac{1}{N}\sum_{i=1}^{N}(H_i - \bar{H})(M_i - \bar{M})$$

http://www.iro.umontreal.ca/~pift6080/H09/documents/papers/pca_tutorial.pdf

PCA:

#3: Eigen-calculation

$$C = \begin{bmatrix} cov(HH) & cov(HM) \\ cov(MH) & cov(MM) \end{bmatrix}$$

$$eigenvalue(C), eigenvector(C)$$

Form the matrix $A - \lambda I$:

$$A - \lambda I = \begin{pmatrix} 1 & -3 & 3 \\ 3 & -5 & 3 \\ 6 & -6 & 4 \end{pmatrix} - \begin{pmatrix} \lambda & 0 & 0 \\ 0 & \lambda & 0 \\ 0 & 0 & \lambda \end{pmatrix} = \begin{pmatrix} 1-\lambda & -3 & 3 \\ 3 & -5-\lambda & 3 \\ 6 & -6 & 4-\lambda \end{pmatrix}$$

PCA new coordinate

SVD (Singular Value Decomposition)

Eigen values

Eigen vectors

$$A \cdot v = \lambda \cdot v$$

$$A \cdot v - \lambda \cdot v = 0$$

$$A \cdot v - \lambda \cdot I \cdot v = 0$$

$$(A - \lambda \cdot I) \cdot v = 0$$

Calculate $\det(A - \lambda I)$:

$$\begin{aligned} \det(A - \lambda I) &= (1-\lambda)\begin{vmatrix} -5-\lambda & 3 \\ -6 & 4-\lambda \end{vmatrix} - (-3)\begin{vmatrix} 3 & 3 \\ 6 & 4-\lambda \end{vmatrix} + 3\begin{vmatrix} 3 & -5-\lambda \\ 6 & -6 \end{vmatrix} \\ &= (1-\lambda)\left((-5-\lambda)(4-\lambda) - (3)(-6)\right) + 3(3(4-\lambda) - 3 \times 6) + 3(3 \times (-6) - (-5-\lambda)6) \\ &= (1-\lambda)(-20 + 5\lambda - 4\lambda + \lambda^2 + 18) + 3(12 - 3\lambda - 18) + 3(-18 + 30 + 6\lambda) \\ &= (1-\lambda)(-2 + \lambda + \lambda^2) + 3(-6 - 3\lambda) + 3(12 + 6\lambda) \\ &= -2 + \lambda + \lambda^2 + 2\lambda - \lambda^2 - \lambda^3 - 18 - 9\lambda + 36 + 18\lambda \\ &= 16 + 12\lambda - \lambda^3. \end{aligned}$$

To find solutions to $\det(A - \lambda I) = 0$ i.e., to solve

$$\lambda^3 - 12\lambda - 16 = 0.$$

http://www.iro.umontreal.ca/~pift6080/H09/documents/papers/pca_tutorial.pdf

PCA:

#3: Eigen-calculation

$$C = \begin{bmatrix} cov(HH) & cov(HM) \\ cov(MH) & cov(MM) \end{bmatrix}$$

$$eigenvalue(C), eigenvector(C)$$

PCA new coordinate

SVD (Singular Value Decomposition)

Eigen values

Eigen vectors

To find solutions to $\det(A - \lambda I) = 0$ i.e., to solve

$$\lambda^3 - 12\lambda - 16 = 0.$$

eigenvalues of A are $\lambda = 4, -2$. ($\lambda = -2$ is a repeated root

Case 1: $\lambda = 4$

$$\left(A - \lambda I \,\vdots\, \mathbf{0}\right)$$

$$\begin{array}{cc} \begin{pmatrix} -3 & -3 & 3 & 0 \\ 3 & -9 & 3 & 0 \\ 6 & -6 & 0 & 0 \end{pmatrix} & \begin{array}{c} \text{R1} \\ \text{R2} \\ \text{R3} \end{array} \end{array} \xrightarrow{R1 \to -1/3 \times R3} \begin{array}{cc} \begin{pmatrix} 1 & 1 & -1 & 0 \\ 3 & -9 & 3 & 0 \\ 6 & -6 & 0 & 0 \end{pmatrix} & \begin{array}{c} \text{R1} \\ \text{R2} \\ \text{R3} \end{array} \end{array}$$

$$\xrightarrow[R3 \to R3 - 6 \times R1]{R2 \to R2 - 3 \times R1} \begin{array}{cc} \begin{pmatrix} 1 & 1 & -1 & 0 \\ 0 & -12 & 6 & 0 \\ 0 & -12 & 6 & 0 \end{pmatrix} & \begin{array}{c} \text{R1} \\ \text{R2} \\ \text{R3} \end{array} \end{array}$$

$$\xrightarrow{R2 \to -1/12 \times R2} \begin{array}{cc} \begin{pmatrix} 1 & 1 & -1 & 0 \\ 0 & 1 & -1/2 & 0 \\ 0 & -12 & 6 & 0 \end{pmatrix} & \begin{array}{c} \text{R1} \\ \text{R2} \\ \text{R3} \end{array} \end{array}$$

$$\xrightarrow{R3 \to R3 + 12 \times R2} \begin{array}{cc} \begin{pmatrix} 1 & 1 & -1 & 0 \\ 0 & 1 & -1/2 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} & \begin{array}{c} \text{R1} \\ \text{R2} \\ \text{R3} \end{array} \end{array}$$

$$\xrightarrow{R1 \to R1 - R2} \begin{array}{cc} \begin{pmatrix} 1 & 0 & -1/2 & 0 \\ 0 & 1 & -1/2 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} & \begin{array}{c} \text{R1} \\ \text{R2} \\ \text{R3} \end{array} \end{array}$$

http://www.iro.umontreal.ca/~pift6080/H09/documents/papers/pca_tutorial.pdf

PCA:

#3: Eigen-calculation

$$C = \begin{bmatrix} cov(HH) & cov(HM) \\ cov(MH) & cov(MM) \end{bmatrix}$$

$$eigenvalue(C), eigenvector(C)$$

PCA new coordinate

- SVD (Singular Value Decomposition)
- Eigen values
- Eigen vectors

To find solutions to $\det(A - \lambda I) = 0$ i.e., to solve

$$\lambda^3 - 12\lambda - 16 = 0.$$

eigenvalues of A are $\lambda = 4, -2$. ($\lambda = -2$ is a repeated root

Case 1: $\lambda = 4$

$$\left(A - \lambda I \,\vdots\, \mathbf{0}\right)$$

$$\xrightarrow{R1 \to R1 - R2} \begin{array}{cl} \begin{pmatrix} 1 & 0 & -1/2 & 0 \\ 0 & 1 & -1/2 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} & \begin{matrix} \mathrm{R1} \\ \mathrm{R2} \\ \mathrm{R3} \end{matrix} \end{array}$$

$$\begin{aligned} x_1 - 1/2x_3 &= 0 \\ x_2 - 1/2x_3 &= 0 \end{aligned}$$

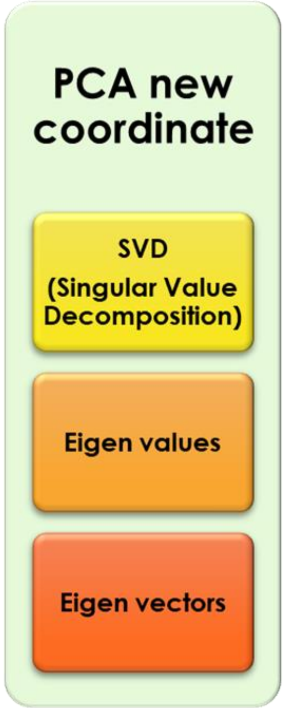
$$\mathbf{x} = \begin{pmatrix} x_1 = \frac{x_3}{2} \\ x_2 = \frac{x_3}{2} \\ x_3 \end{pmatrix} = x_3 \begin{pmatrix} \frac{1}{2} \\ \frac{1}{2} \\ 1 \end{pmatrix}$$

http://www.iro.umontreal.ca/~pift6080/H09/documents/papers/pca_tutorial.pdf

PCA:
#3: Eigen-calculation

$$\boldsymbol{C} = \begin{bmatrix} \boldsymbol{cov(HH)} & \boldsymbol{cov(HM)} \\ \boldsymbol{cov(MH)} & \boldsymbol{cov(MM)} \end{bmatrix}$$

$$\boldsymbol{eigenvalue(C), eigenvector(C)}$$



To find solutions to $\det(A - \lambda I) = 0$ i.e., to solve

$$\lambda^3 - 12\lambda - 16 = 0.$$

eigenvalues of A are $\lambda = 4, -2$. ($\lambda = -2$ is a repeated root

Case 2: $\lambda = -2$

$$\left(A - \lambda I \,\vdots\, \mathbf{0}\right)$$

$$\begin{array}{c} \begin{pmatrix} 3 & -3 & 3 & 0 \\ 3 & -3 & 3 & 0 \\ 6 & -6 & 6 & 0 \end{pmatrix} \end{array} \begin{array}{l} \mathrm{R1} \\ \mathrm{R2} \\ \mathrm{R3} \end{array} \xrightarrow{R1 \to 1/3 \times R3} \begin{pmatrix} 1 & -1 & 1 & 0 \\ 3 & -3 & 3 & 0 \\ 6 & -6 & 6 & 0 \end{pmatrix} \begin{array}{l} \mathrm{R1} \\ \mathrm{R2} \\ \mathrm{R3} \end{array}$$

$$\xrightarrow[R3 \to R3 - 6 \times R1]{R2 \to R2 - 3 \times R1} \begin{pmatrix} 1 & 1 & -1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \begin{array}{l} \mathrm{R1} \\ \mathrm{R2} \\ \mathrm{R3} \end{array}$$

$$x_1 + x_2 - x_3 = 0,$$

$$\mathbf{x} = \begin{pmatrix} x_1 = x_3 - x_2 \\ x_2 \\ x_3 \end{pmatrix}$$

$$\mathbf{x} = \begin{pmatrix} x_3 - x_2 \\ x_2 \\ x_3 \end{pmatrix} = x_3 \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix} + x_2 \begin{pmatrix} -1 \\ 1 \\ 0 \end{pmatrix} \quad \text{for any } x_2, x_3 \in \mathbb{R} \backslash \{0\}$$

http://www.iro.umontreal.ca/~pift6080/H09/documents/papers/pca_tutorial.pdf

PCA:
#3: Eigen-decomposition

$$C = \begin{bmatrix} cov(HH) & cov(HM) \\ cov(MH) & cov(MM) \end{bmatrix}$$

$$eigenvalue(C), eigenvector(C)$$

Sorted eigen_values

[377.32003303,   5.66607808]

Corresponding eigen_vectors of sorted eigen_values

[[-0.32008244, -0.94738969],
 [-0.94738969, 0.32008244]]

Organize PCA dominant dimension

Sorted

Eigen values

Eigen vectors

```
eig_vals, eig_vecs = np.linalg.eig(C)
print('Eigenvectors \n%s' %eig_vecs)
print('\nEigenvalues \n%s' %eig_vals)
```

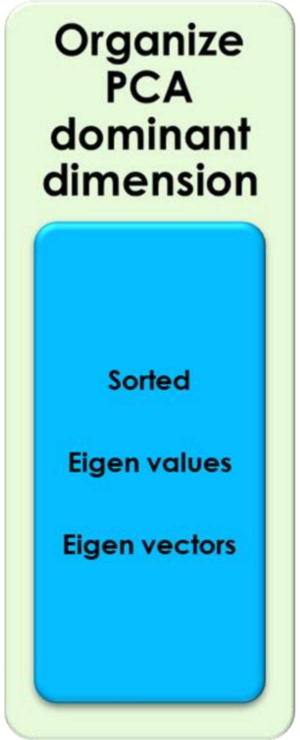
```
Eigenvectors
[[-0.94738969 -0.32008244]
 [ 0.32008244 -0.94738969]]

Eigenvalues
[   5.66607808 377.32003303]
```

Eigenvalues and eigenvectors of C
- is learned (fitted) from data (H,M)
- cannot be controlled
- sometimes resulted in unsolvable eigenvalue

http://www.iro.umontreal.ca/~pift6080/H09/documents/papers/pca_tutorial.pdf

PCA:
#3: Eigen-calculation



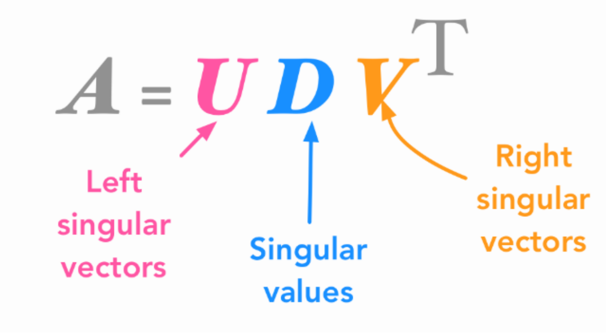
$$C = \begin{bmatrix} cov(HH) & cov(HM) \\ cov(MH) & cov(MM) \end{bmatrix}$$

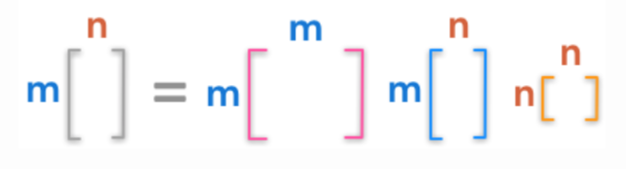
V: $eigenvalue(C^T C), eigenvector(C^T C)$
U: $eigenvalue(CC^T), eigenvector(CC^T)$

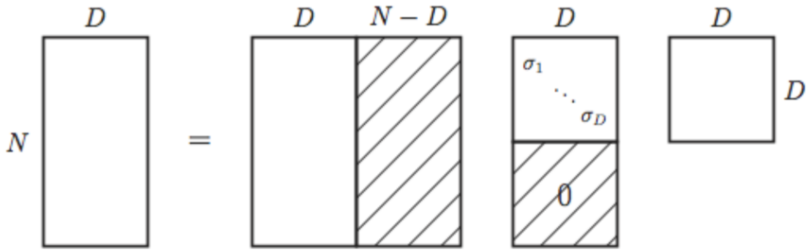
$eigenvalue$ = diagonal (D)

$eigenvector$ = column vector (U)

Through SVD (Singular Value Decomposition)







http://www.iro.umontreal.ca/~pift6080/H09/documents/papers/pca_tutorial.pdf

PCA:
#3: Eigen-calculation

Organize PCA dominant dimension

Sorted

Eigen values

Eigen vectors

$$C = \begin{bmatrix} cov(HH) & cov(HM) \\ cov(MH) & cov(MM) \end{bmatrix}$$

$$eigenvalue(C^T C), eigenvector(C^T C)$$

$eigenvalue$ = diagonal (D)

$eigenvector$ = column vector (U)

Through SVD (Singular Value Decomposition)

```
u,s,v = np.linalg.svd(C)
print('\nEigenvalues \n%s\n' %s)
print('Eigenvectors_u \n%s\n' %u)
print('Eigenvectors_v \n%s\n' %v)
```

```
Eigenvalues
[377.32003303   5.66607808]

Eigenvectors_u
[[-0.32008244 -0.94738969]
 [-0.94738969  0.32008244]]

Eigenvectors_v
[[-0.32008244 -0.94738969]
 [-0.94738969  0.32008244]]
```

PCA:

#4: Selecting and transforming to principle component

Through SVD (Singular Value Decomposition)

Selected dominant eigen	PCA Transform
Selected k-Eigen values Corresponding k-Eigen vectors	Dot_product (Input, k-Eigenvectors) PCA.fit()

K = 1

```
u,s,v = np.linalg.svd(C)
print('\nEigenvalues \n%s\n' %s)
print('Eigenvectors_u \n%s\n' %u)
print('Eigenvectors_v \n%s\n' %v)
```

```
Eigenvalues
[377.32003303   5.66607808]

Eigenvectors_u
[[-0.32008244 -0.94738969]
 [-0.94738969  0.32008244]]

Eigenvectors_v
[[-0.32008244 -0.94738969]
 [-0.94738969  0.32008244]]
```

PCA(i) = X (i) . [Eigen vector (pca $_i$)]

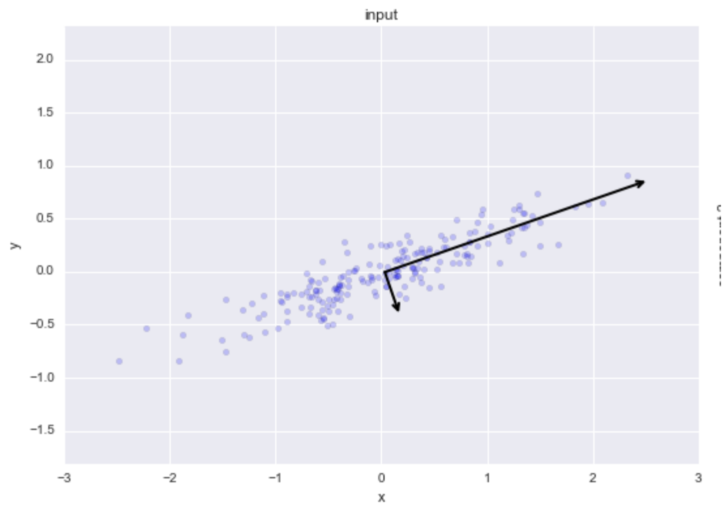
10x1 10x2 2x1

	H	M
1	9	39
2	15	56
3	25	93
4	14	61
5	10	50
6	18	75
7	0	32
8	16	85
9	5	42
10	19	70

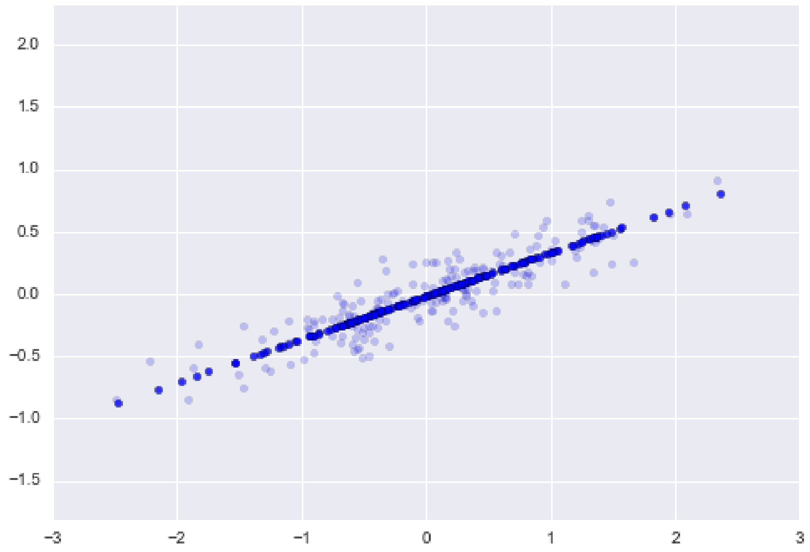
```
pca1
[[-0.32008244]
 [-0.94738969]]
```

PCA:
#4: Selecting and transforming to principle component

Through SVD (Singular Value Decomposition)



X-Y coordinate



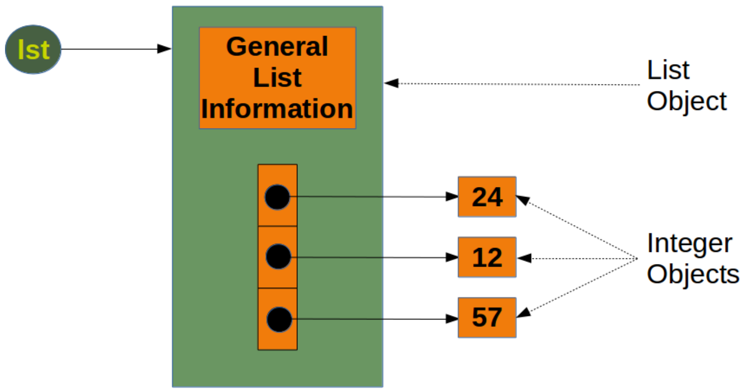
Transformed on to pca1

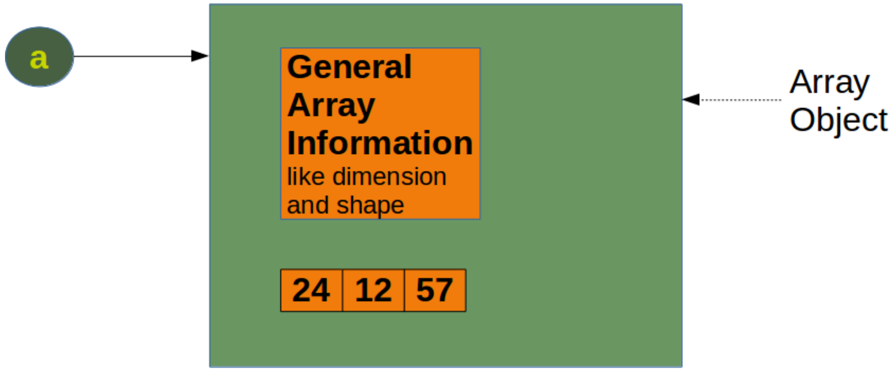
Organizing Data

How would we arrange data for the model?

Is the data in correct format?

Numpy array vs list





- Numpy Array
  - Numpy data structures perform better in:
  - Size - less space
  - Performance faster speed
  - Functionality
    - optimize for math functions

Lists

Every new element
-> additional 8 bytes
for the reference to new object.
-> The integer object itself
consumes 28 bytes.

Data Structure

- ❑ **How can we convert from Pandas dataframe to numpy array**

```
import pandas as pd
import numpy as np
```

```
df = pd.DataFrame({'a':[21, 23, 32, 52],
                   'b':[72, 78, 74, 54],
                   'c':[67.1, 69.5, 56.6, 76.2]})
df
```

	a	b	c
0	21	72	67.1
1	23	78	69.5
2	32	74	56.6
3	52	54	76.2

```
df.shape
```

```
(4, 3)
```

```
df.dtypes
```

```
age          int64
height       int64
weight       int64
dtype: object
```

- https://webcourses.ucf.edu/courses/1249560/pages/python-lists-vs-numpy-arrays-what-is-the-difference?source=post_page-----f1b582a413f1----------------------

Data Structure

- **From Pandas dataframe to numpy array**

	a	b	c
0	21	72	67.1
1	23	78	69.5
2	32	74	56.6
3	52	54	76.2

df.values( )   vs df.to_numpy( ) ????

- https://webcourses.ucf.edu/courses/1249560/pages/python-lists-vs-numpy-arrays-what-is-the-difference?source=post_page-----f1b582a413f1---------------------

Data Structure

❑ From Pandas dataframe to numpy array

	a	b	c
0	21	72	67.1
1	23	78	69.5
2	32	74	56.6
3	52	54	76.2

df.values( ) vs df.to_numpy( ) ????

```
x=df.to_numpy()
x
```

```
array([[21. , 72. , 67.1],
       [23. , 78. , 69.5],
       [32. , 74. , 56.6],
       [52. , 54. , 76.2]])
```

```
df.values
```

```
array([[21. , 72. , 67.1],
       [23. , 78. , 69.5],
       [32. , 74. , 56.6],
       [52. , 54. , 76.2]])
```

- https://webcourses.ucf.edu/courses/1249560/pages/python-lists-vs-numpy-arrays-what-is-the-difference?source=post_page-----f1b582a413f1---------------------

Data Structure

❑ From Pandas dataframe to numpy array

	a	b	c
0	21	72	67.1
1	23	78	69.5
2	32	74	56.6
3	52	54	76.2

df.transpose( ) vs df.values.transpose( ) ????

- https://webcourses.ucf.edu/courses/1249560/pages/python-lists-vs-numpy-arrays-what-is-the-difference?source=post_page-----f1b582a413f1----------------------

Data Structure

❑ From Pandas dataframe to numpy array

	a	b	c
0	21	72	67.1
1	23	78	69.5
2	32	74	56.6
3	52	54	76.2

df.transpose( ) vs df.values.transpose( ) ????

```
df.transpose()
```

	0	1	2	3
a	21.0	23.0	32.0	52.0
b	72.0	78.0	74.0	54.0
c	67.1	69.5	56.6	76.2

```
df.values.transpose()
```

```
array([[21. , 23. , 32. , 52. ],
       [72. , 78. , 74. , 54. ],
       [67.1, 69.5, 56.6, 76.2]])
```

- https://webcourses.ucf.edu/courses/1249560/pages/python-lists-vs-numpy-arrays-what-is-the-difference?source=post_page-----f1b582a413f1---------------------

Data Structure

❑ **From Pandas dataframe to numpy array**

	a	b	c
0	21	72	67.1
1	23	78	69.5
2	32	74	56.6
3	52	54	76.2

df.values.transpose( ) vs df.values.reshape ( ) ????

- https://webcourses.ucf.edu/courses/1249560/pages/python-lists-vs-numpy-arrays-what-is-the-difference?source=post_page-----f1b582a413f1---------------------

Data Structure

❑ From Pandas dataframe to numpy array

	a	b	c
0	21	72	67.1
1	23	78	69.5
2	32	74	56.6
3	52	54	76.2

df.values.transpose( ) vs df.values.reshape ( ) ????

```
df.values.transpose()
```

```
array([[21. , 23. , 32. , 52. ],
       [72. , 78. , 74. , 54. ],
       [67.1, 69.5, 56.6, 76.2]])
```

```
df.values.reshape(3,4)
```

```
array([[21. , 72. , 67.1, 23. ],
       [78. , 69.5, 32. , 74. ],
       [56.6, 52. , 54. , 76.2]])
```

- https://webcourses.ucf.edu/courses/1249560/pages/python-lists-vs-numpy-arrays-what-is-the-difference?source=post_page-----f1b582a413f1---------------------

Data Structure

❑ From Pandas dataframe to numpy array

	a	b	c
0	21	72	67.1
1	23	78	69.5
2	32	74	56.6
3	52	54	76.2

df.values.transpose( ) vs df.values.reshape ( ) ????

```
df.values.transpose()
```

```
array([[21. , 23. , 32. , 52. ],
       [72. , 78. , 74. , 54. ],
       [67.1, 69.5, 56.6, 76.2]])
```

```
df.values.reshape(3,4,order='C')
```

```
array([[21. , 72. , 67.1, 23. ],
       [78. , 69.5, 32. , 74. ],
       [56.6, 52. , 54. , 76.2]])
```

```
df.values.reshape(3,4,order='F')
```

```
array([[21. , 52. , 74. , 69.5],
       [23. , 72. , 54. , 56.6],
       [32. , 78. , 67.1, 76.2]])
```

- https://webcourses.ucf.edu/courses/1249560/pages/python-lists-vs-numpy-arrays-what-is-the-difference?source=post_page-----f1b582a413f1---------------------

Data Structure

❑ **Reshape 2D to 3D ????**

	a	b	c
0	21	72	67.1
1	23	78	69.5
2	32	74	56.6
3	52	54	76.2

df.values.reshape(1,3,4)   vs df.values.reshape(3,4,1)

- https://webcourses.ucf.edu/courses/1249560/pages/python-lists-vs-numpy-arrays-what-is-the-difference?source=post_page-----f1b582a413f1----------------------

Data Structure

- ❑ **Reshape 2D to 3D ????**

	a	b	c
0	21	72	67.1
1	23	78	69.5
2	32	74	56.6
3	52	54	76.2

df.values.reshape(1,3,4) vs df.values.reshape(3,4,1)

```
df.values.reshape(3,4,1)
```

```
array([[[21. ],
        [72. ],
        [67.1],
        [23. ]],

       [[78. ],
        [69.5],
        [32. ],
        [74. ]],

       [[56.6],
        [52. ],
        [54. ],
        [76.2]]])
```

```
df.values.reshape(1,3,4)
```

```
array([[[21. , 72. , 67.1, 23. ],
        [78. , 69.5, 32. , 74. ],
        [56.6, 52. , 54. , 76.2]]])
```

- https://webcourses.ucf.edu/courses/1249560/pages/python-lists-vs-numpy-arrays-what-is-the-difference?source=post_page-----f1b582a413f1---------------------

Data Structure

- ❑ **Time-series split**

	a	b	c
0	21	72	67.1
1	23	78	69.5
2	32	74	56.6
3	52	54	76.2

Time step = 2
Time stride = 1

	a	b	c
0	21	72	67.1
1	23	78	69.5
1	23	78	69.5
2	32	74	56.6
2	32	74	56.6
3	52	54	76.2

.shape = ???

Activity: Data Preparation

❑ Time-series split

No	uts	Time Diff	x	y	z	Type A	Type B
1	2019-01-17 09:14:17+07:00		2.729	-3.035	8.684	1	1
2	2019-01-17 09:14:38+07:00		2.586	-2.633	500	1	1
3	2019-01-17 09:15:02+07:00		1.67	-2.117	8.512	0	0
4	2019-01-17 09:15:26+07:00			-9.788	2.519	0	1
5	2019-01-17 09:15:48+07:00		0.241		3.03	1	0
6	2019-01-17 09:16:08+07:00		1.78		3.114	1	0
7	2019-01-17 09:16:59+07:00		1.823		4.414	0	0
8	2019-01-17 09:17:23+07:00		0.103	-8.909	5.472	0	0
9	2019-01-17 09:17:44+07:00		2.046	-2.218	8.572	0	1
10	2019-01-17 09:18:05+07:00		2.28	-2.421	8.761	0	1

1. Clean / Combine Data

2. Calculate Time diff

   What would be a problem?

3. What would be the shape for

   Time-series split with

   Time step = 3 / Time stride = 2?

   (#sample, #time_step, #Feature)