L14: Dimension Reduction

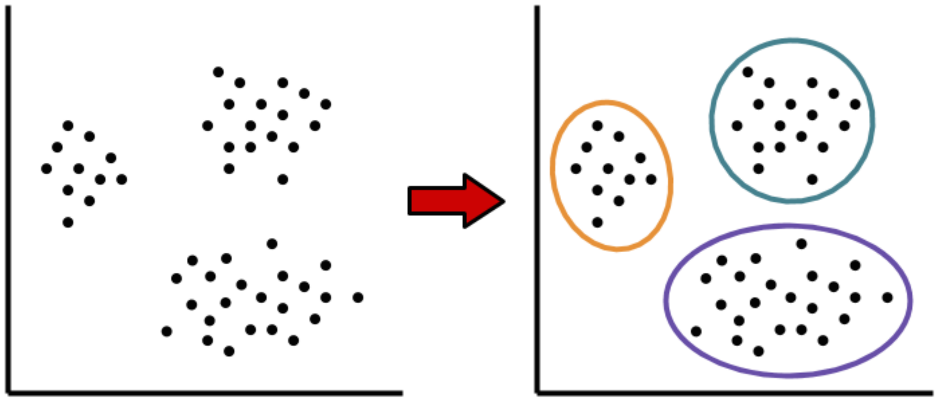
Shan Wang

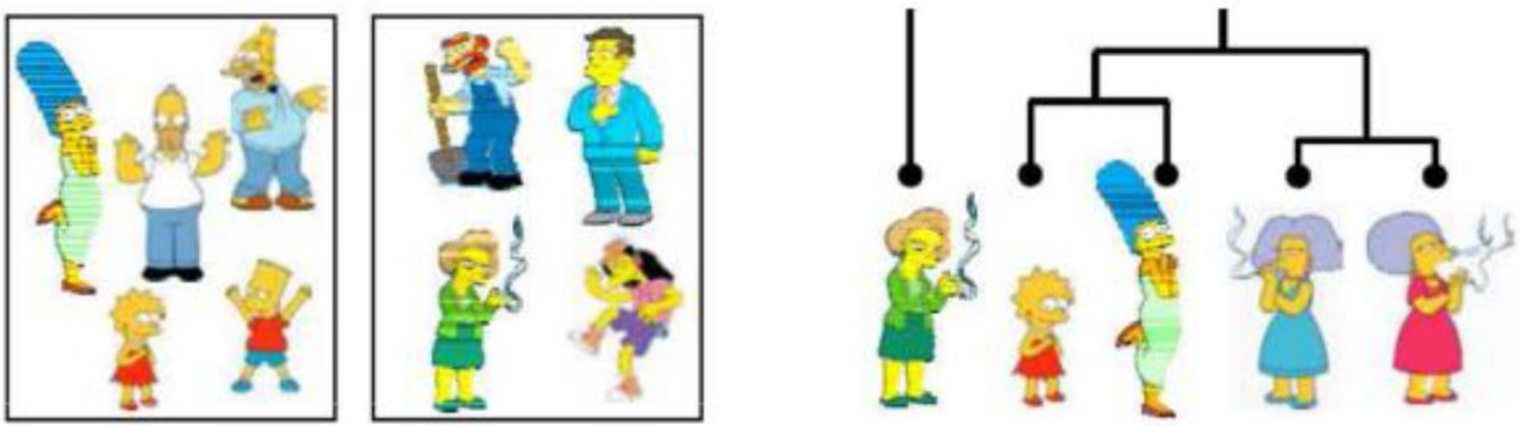
Lingnan College, Sun Yat-sen University

2020 Data Mining and Machine Learning LN3119
https://wangshan731.github.io/DM-ML/

Last lecture

- Unsupervised Learning
- Clustering
  - Hierarchical clustering
  - k-means clustering
- Applications: Netflix





Course Outline

- Supervised learning
  - Linear regression
  - Logistic regression
  - SVM and kernel
  - Tree models
- Unsupervised learning
  - Clustering
  - PCA (Dimension Reduction)
  - EM
- Deep learning
  - Neural networks
  - Convolutional NN
  - Recurrent NN
- Reinforcement learning
  - MDP
  - ADP
  - Deep Q-Network

This lecture

- Dimension Reduction
  - Motivation
  - PCA
  - SVD
  - Autoencoder revisit
  - PCA lab
- EM method

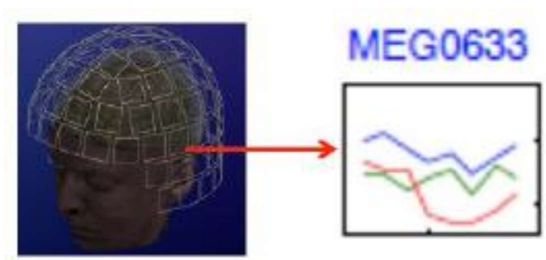
Reference: CS 420, Weinan Zhang (SJTU)

Motivation

- Suppose we want to predict the health condition of some students, and the features for the students includes:
  - Weight in kilogram
  - Height in inch
  - Height in cm
  - Hours of sports per day
  - Favorite color
  - Favorite color
- Some features are irrelevant, e.g. favorite color and scores in math
- Some features are redundant, e.g. height in inch and cm

High dimensional data

- In the era of big data, the dimensionality increases dramatically
    - E.g. there are many features for the electroencephalogram data



- It becomes very important to reduce the dimensionality, or select the most important features, or find the most representative features

Principal Components Analysis

- Principal components analysis (PCA) is a technique that can be used to simplify a dataset
- It is usually a linear transformation that chooses a new coordinate system for the data set such that
  - greatest variance by any projection of the dataset comes to lie on the first axis (then called the first principal component)
  - the second greatest variance on the second axis, and so on
- PCA can be used for reducing dimensionality by eliminating the later principal components

Example

- Consider the following 3D points

1	2	4	3	5	6
2	4	8	6	10	12
3	6	12	9	15	8

- If each component is stored in a byte, we need 18 = 3 × 6 bytes

Example (cont.)

- Looking closer, we can see that all the points are related geometrically
  - they are all in the same direction, scaled by a factor:

$$\begin{bmatrix}1\\2\\3\end{bmatrix} = 1 \times \begin{bmatrix}1\\2\\3\end{bmatrix} \qquad \begin{bmatrix}4\\8\\12\end{bmatrix} = 4 \times \begin{bmatrix}1\\2\\3\end{bmatrix} \qquad \begin{bmatrix}5\\10\\15\end{bmatrix} = 5 \times \begin{bmatrix}1\\2\\3\end{bmatrix}$$

$$\begin{bmatrix}2\\4\\6\end{bmatrix} = 2 \times \begin{bmatrix}1\\2\\3\end{bmatrix} \qquad \begin{bmatrix}3\\6\\9\end{bmatrix} = 3 \times \begin{bmatrix}1\\2\\3\end{bmatrix} \qquad \begin{bmatrix}6\\12\\8\end{bmatrix} = 6 \times \begin{bmatrix}1\\2\\3\end{bmatrix}$$

Example (cont.)

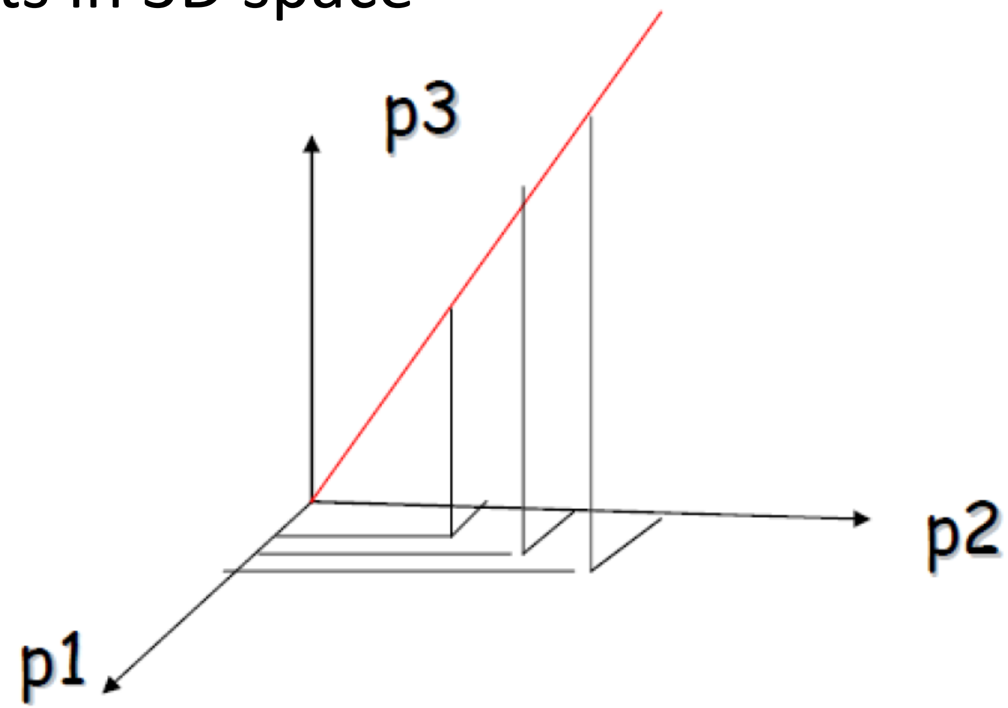
$$\begin{bmatrix}1\\2\\3\end{bmatrix} = 1 \times \begin{bmatrix}1\\2\\3\end{bmatrix} \quad \begin{bmatrix}4\\8\\12\end{bmatrix} = 4 \times \begin{bmatrix}1\\2\\3\end{bmatrix} \quad \begin{bmatrix}5\\10\\15\end{bmatrix} = 5 \times \begin{bmatrix}1\\2\\3\end{bmatrix}$$

$$\begin{bmatrix}2\\4\\6\end{bmatrix} = 2 \times \begin{bmatrix}1\\2\\3\end{bmatrix} \quad \begin{bmatrix}3\\6\\9\end{bmatrix} = 3 \times \begin{bmatrix}1\\2\\3\end{bmatrix} \quad \begin{bmatrix}6\\12\\8\end{bmatrix} =6 \times \begin{bmatrix}1\\2\\3\end{bmatrix}$$

- They can be stored using only 9 bytes (50% savings!):
  - Store one direction (3 bytes) + the multiplying constants (6 bytes)

Geometrical interpretation

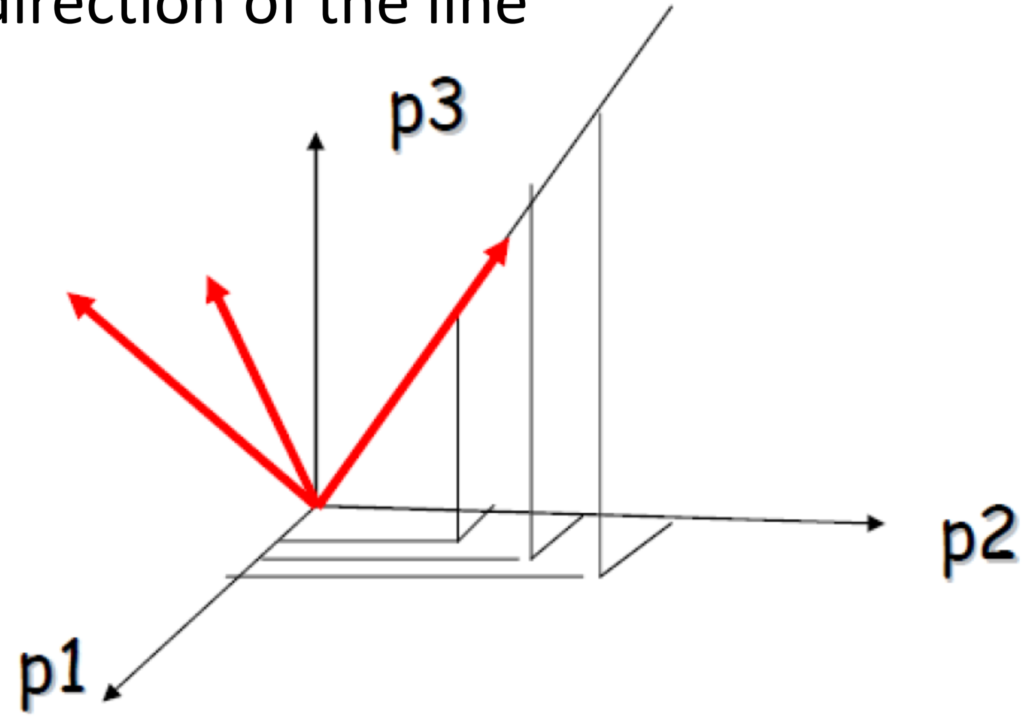
- View points in 3D space



- In this example, all the points happen to lie on one line
    - a 1D subspace of the original 3D space

Geometrical interpretation

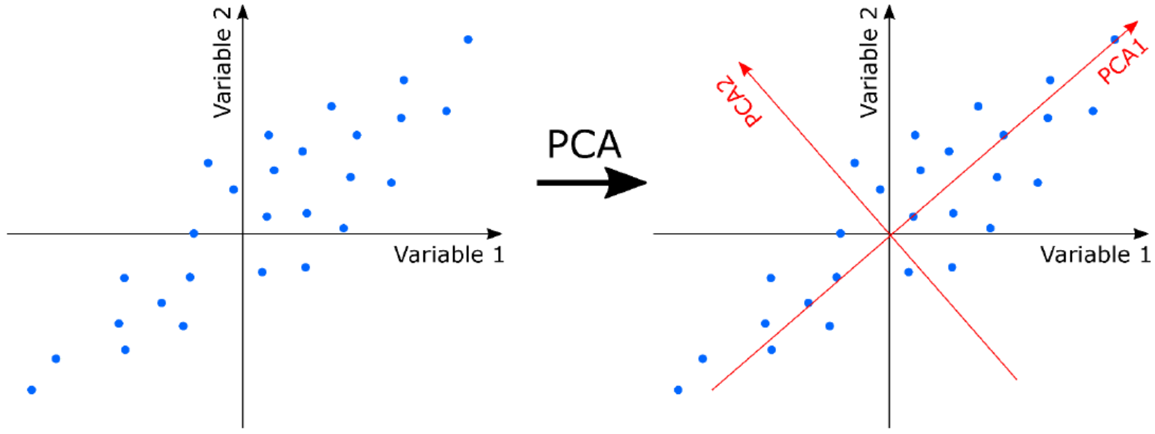
- Consider a new coordinate system where the first axis is along the direction of the line



- In the new coordinate system, every point has only one non-zero coordinate
  - we only need to store the direction of the line (a 3 bytes point) and the nonzero coordinates for each point (6 bytes)

Back to PCA

- Given a set of points, how can we know if they can be compressed similarly to the previous example?
  - We can look into the correlation between the points by the tool of PCA

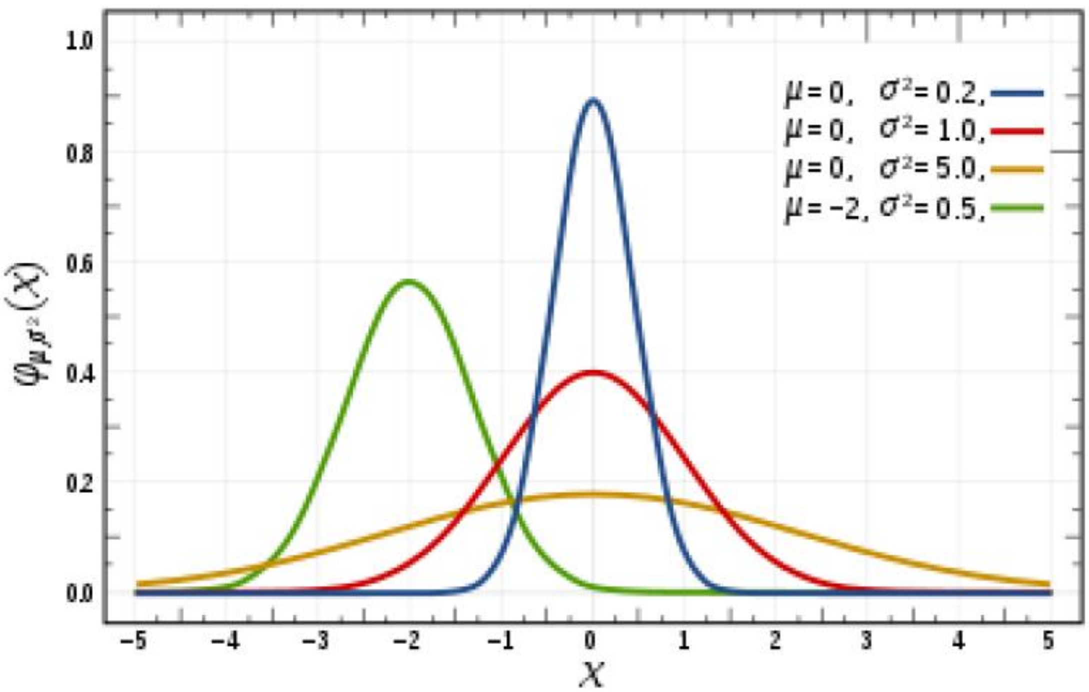


From example to theory

- In previous example, PCA rebuilds the coordination system for the data by selecting
  - the direction with largest variance as the first new base direction
  - the direction with the second largest variance as the second new base direction
  - and so on
- Then how can we find the direction with largest variance?
  - By the eigenvector for the covariance matrix of the data

Review – Variance

- Variance is the expectation of the squared deviation of a random variable from its mean
  - Informally, it measures how far a set of (random) numbers are spread out from their average value

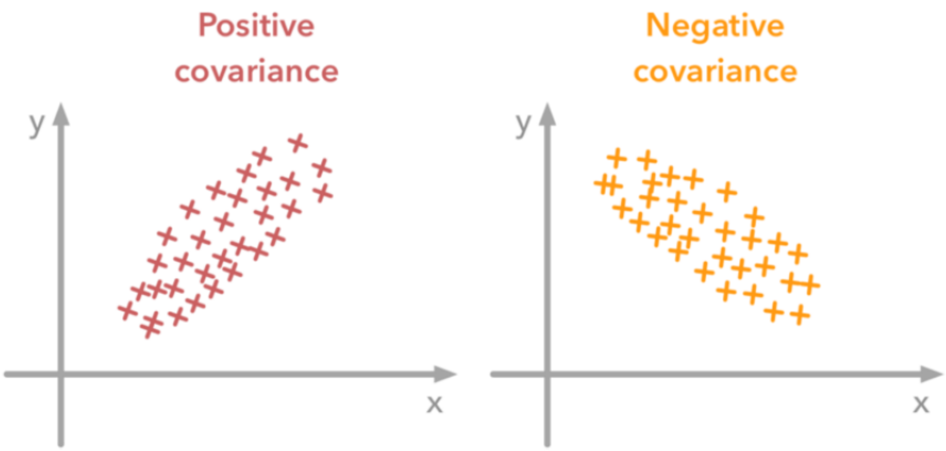


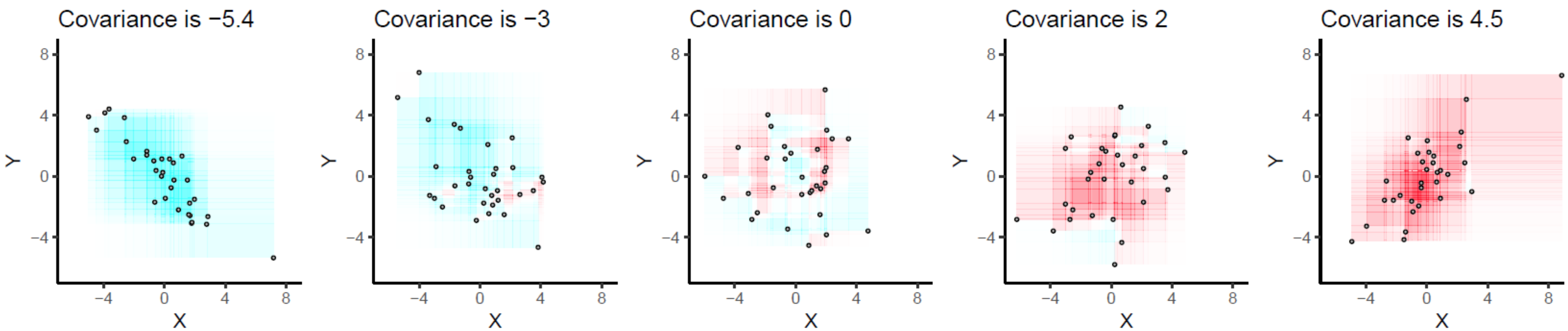
Review – Covariance

- Covariance is a measure of the joint variability of two random variables
  - If the greater values of one variable mainly correspond with the greater values of the other variable, (i.e., the variables tend to show similar behavior), the covariance is positive
    - E.g. as the number of hours studied increases, the marks in that subject increase
  - In the opposite case, the covariance is negative
  - The sign of the covariance therefore shows the tendency in the linear relationship between the variables
  - The magnitude of the covariance is not easy to interpret. The normalized version of the covariance, the correlation coefficient, however, shows by its magnitude the strength of the linear relation

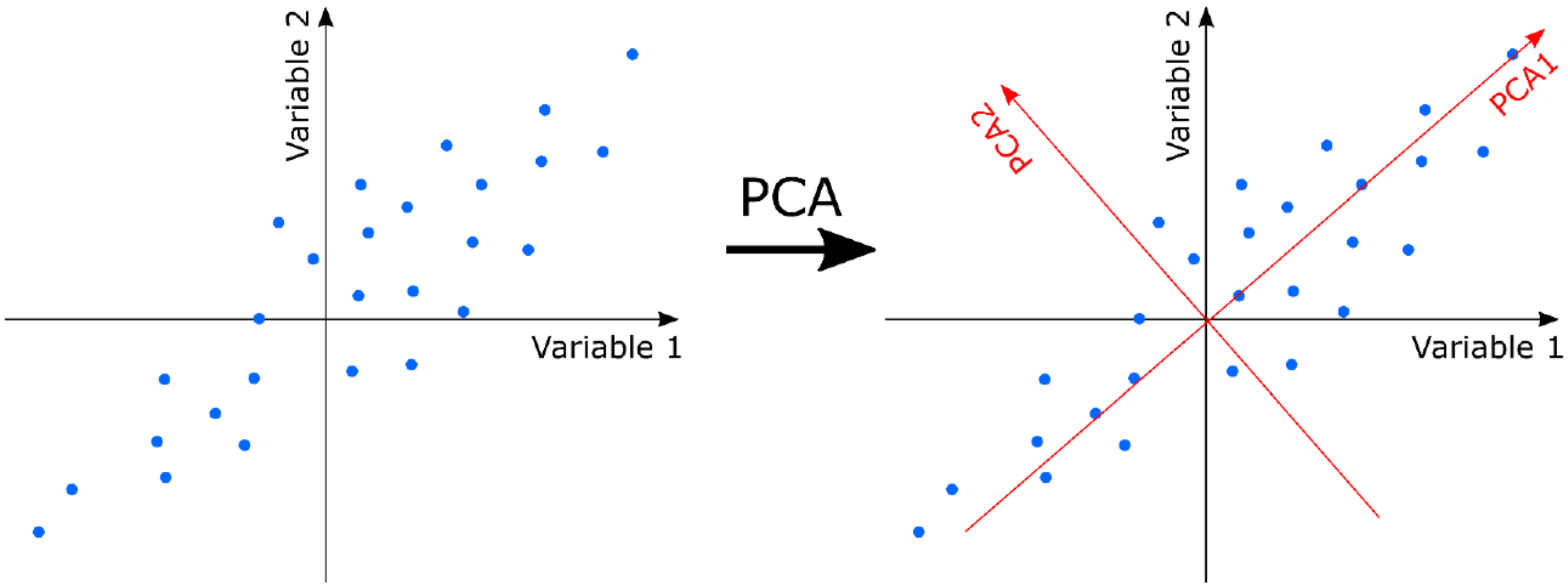
Review – Covariance (cont.)

- $covariance(X, Y) = \frac{\sum_{i=1}^{N}(x_i - \bar{x})(y_i - \bar{y})}{N-1}$

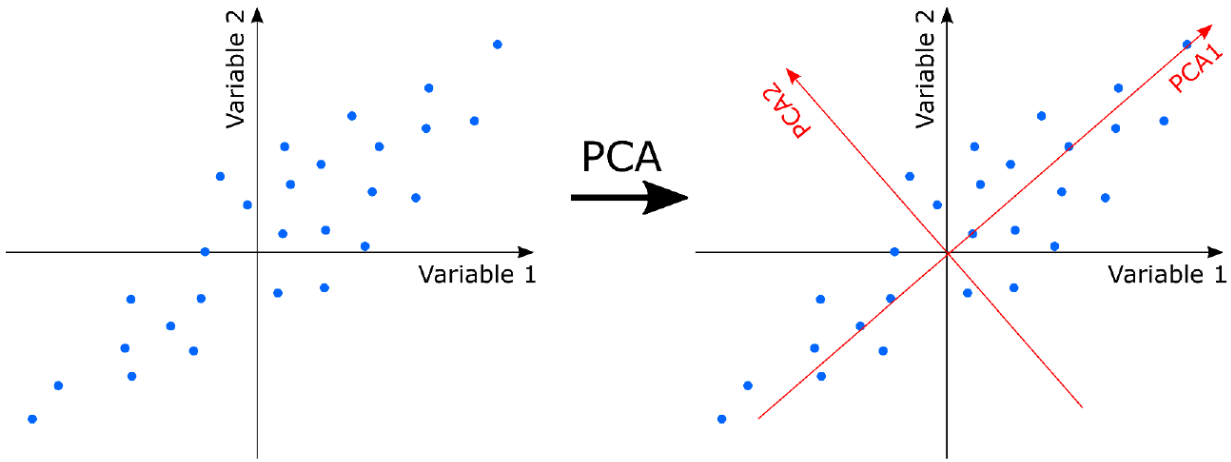




PCA



PCA



- PCA tries to identify the subspace in which the data approximately lies in
- PCA uses an orthogonal transformation on the coordinate system to convert a set of observations of possibly correlated variables into a set of values of linearly uncorrelated variables called principal components
  - The number of principal components is less than or equal to $\min\{d, N\}$

Covariance matrix

- Suppose there are 3 dimensions, denoted as X, Y, Z. The covariance matrix is

$$COV = \begin{bmatrix} COV(X,X) & COV(X,Y) & COV(X,Z) \\ COV(Y,X) & COV(Y,Y) & COV(Y,Z) \\ COV(Z,X) & COV(Z,Y) & COV(Z,Z) \end{bmatrix}$$

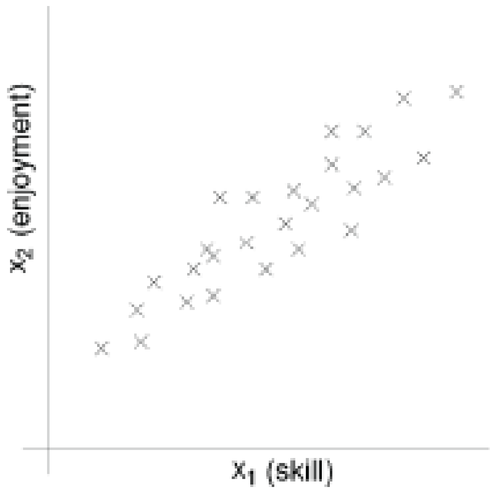
- Note the diagonal is the covariance of each dimension with respect to itself, which is just the variance of each random variable
- Also $COV(X, Y) = COV(Y, X)$
  - hence matrix is symmetric about the diagonal
- d-dimensional data will result in a $d \times d$ covariance matrix

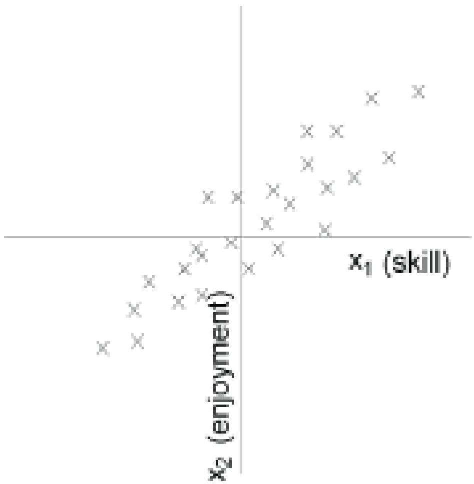
Covariance in the covariance matrix

- Diagonal, or the variance, measures the deviation from the mean for data points in one dimension
- Covariance measures how one dimension random variable varies w.r.t. another, or if there is some linear relationship among them

Data processing

- Given the dataset $D = \{x^{(i)}\}_{i=1}^{N}$
- Let $\bar{x} = \frac{1}{N}\sum_{i=1}^{N} x^{(i)}$
- $X = \begin{bmatrix} (x^{(1)} - \bar{x})' \\ \vdots \\ (x^{(i)} - \bar{x})' \\ (x^{(N)} - \bar{x})' \end{bmatrix}$





- Move the center of the data set to 0

Data processing (cont.)

- $Q = X'X =$

$$[(x^{(1)} - \bar{x})' \cdots (x^{(i)} - \bar{x})' (x^{(N)} - \bar{x})'] \begin{bmatrix} (x^{(1)} - \bar{x})' \\ \vdots \\ (x^{(i)} - \bar{x})' \\ (x^{(N)} - \bar{x})' \end{bmatrix}$$

- Q is square with d dimension
- Q is symmetric
- Q is the covariance matrix [aka scatter matrix]
- Q can be very large (in vision, d is often the number of pixels in an image!)
  - or a 256×256 image, d = 65536!!
  - Don't want to explicitly compute Q

PCA

- By finding the eigenvalues and eigenvectors of the covariance matrix, we find that the eigenvectors with the largest eigenvalues correspond to the dimensions that have the strongest variation in the dataset
- This is the principal component

Eigenvector of Matrix A

$$\mathbf{Ax} = \lambda\mathbf{x}$$

Eigenvalue of Matrix A

- Application:
  - face recognition, image compression
  - finding patterns in data of high dimension

PCA theorem

- Theorem:
- Each $x^{(i)}$ can be written as: $x^{(i)} = \bar{x} + \sum_{j=1}^{d} g_{ij} e_j$ where e_j are the d eigenvectors of Q with non-zero eigenvalues

- Notes:
  1. The eigenvectors $e_1, e_2, \dots, e_d$ span an eigenspace
  2. $e_1, e_2, \dots, e_d$ are $d \times 1$ orthonormal vectors (directions in d-Dimensional space)
  3. The scalars g_{ij} are the coordinates of $x^{(i)}$ in the space

$$g_{ij} = < x^{(i)} - \bar{x}, e_j >$$

Using PCA to compress data

- Expressing x in terms of $e_1, e_2, \dots, e_d$ doesn't change the size of the data
- Sort the eigenvectors e_j according to their eigenvalue

$$\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_d$$

- Assume $\lambda_j \approx 0$ if $j > k$. Then

$$x^{(i)} \approx \bar{x} + \sum_{j=1}^{k} g_{ij} e_j$$

Bonus question:

$$\lambda_1 + \lambda_2 + \cdots + \lambda_d = ?$$

How to solve eigenvalues and eigenvectors?

- If $(A - \lambda I)x = 0$ has a nonzero solution for λ, then $A - \lambda I$ is not invertible. Then the determinant of $A - \lambda I$ must be zero

- λ is an eigenvalue of A if and only if $A - \lambda I$ is singular:

$$\det(A - \lambda I) = 0$$

$$\mathbf{A}\overbrace{\mathbf{x}}^{\text{Eigenvector of Matrix A}} = \underbrace{\lambda}_{\text{Eigenvalue of Matrix A}}\overbrace{\mathbf{x}}^{\text{Eigenvector of Matrix A}}$$

Example

- Find the eigenvalues and eigenvectors of $A = \begin{bmatrix} 1 & 2 \\ 2 & 4 \end{bmatrix}$
- Need to solve $\det(A - \lambda I) = 0$
- $A - \lambda I = \begin{bmatrix} 1 - \lambda & 2 \\ 2 & 4 - \lambda \end{bmatrix}$
- $\det(A - \lambda I) = (1 - \lambda)(4 - \lambda) - 2 \times 2 = \lambda^2 - 5\lambda$
- $\det(A - \lambda I) = 0$ would imply $\lambda = 0$ and $\lambda = 5$

Example (cont.)

- hen solve $(A - \lambda I)\,x = 0$ to get the eigenvectors for each of the eigenvalues
- $Ax = 0$ has a nonzero solution of $\begin{bmatrix} 2 \\ -1 \end{bmatrix}$ or $\begin{bmatrix} 2/\sqrt{5} \\ -1/\sqrt{5} \end{bmatrix}$
- $(A - 5I)x = 0$ has a nonzero solution of $\begin{bmatrix} 1 \\ 2 \end{bmatrix}$ or $\begin{bmatrix} 1/\sqrt{5} \\ 2/\sqrt{5} \end{bmatrix}$
- Note: Eigenvectors for different eigenvalues are orthogonal

How about non-squared matrix A?

SVD

Singular Value Decomposition

SVD

- Singular Value Decomposition (SVD) is a factorization method of matrix. It states that any $m \times n$ matrix A can be written as the product of 3 matrices:

$$A = USV^T$$

- Where:
  - U is $m \times m$ and its columns are orthonormal eigenvectors of AA^T
  - V is $n \times n$ and its columns are orthonormal eigenvectors of A^TA
  - S is $m \times n$ is a diagonal matrix with r elements equal to the root of the positive eigenvalues of AA^T or A^TA (both matrices have the same positive eigenvalues anyway)

In full matrix form

$$A_{m\times n} = U_{m\times m} S_{m\times n} V^T_{n\times n}$$

$$\underset{m\times n}{\overset{\mathrm{A}}{\begin{pmatrix} x_{11} & x_{12} & & x_{1n} \\ \vdots & & \ddots & \\ x_{m1} & & & x_{mn} \end{pmatrix}}} = \underset{m\times m}{\overset{\mathrm{U}}{\begin{pmatrix} u_{11} & & u_{m1} \\ & \ddots & \\ u_{1m} & & u_{mm} \end{pmatrix}}} \underset{m\times n}{\overset{\mathrm{S}}{\begin{pmatrix} \sigma_1 & & & 0 \\ & \ddots & & \\ & & \sigma_r & \\ & & & \ddots \\ 0 & & & 0 \end{pmatrix}}} \underset{n\times n}{\overset{\mathrm{V^T}}{\begin{pmatrix} v_{11} & & v_{1n} \\ & \ddots & \\ v_{n1} & & v_{nn} \end{pmatrix}}}$$

Example

- Let's assume：

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

- We can have:

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

Example (cont.)

- Compute U and V respectively:

$$AA^T = \begin{pmatrix} 17 & 8 \\ 8 & 17 \end{pmatrix}$$

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

eigenvectors

$$u_1 = \begin{pmatrix} 1/\sqrt{2} \\ 1/\sqrt{2} \end{pmatrix} \quad u_2 = \begin{pmatrix} 1/\sqrt{2} \\ -1/\sqrt{2} \end{pmatrix}$$

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

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

eigenvectors

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

Example (cont.)

- Finally, we have:

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

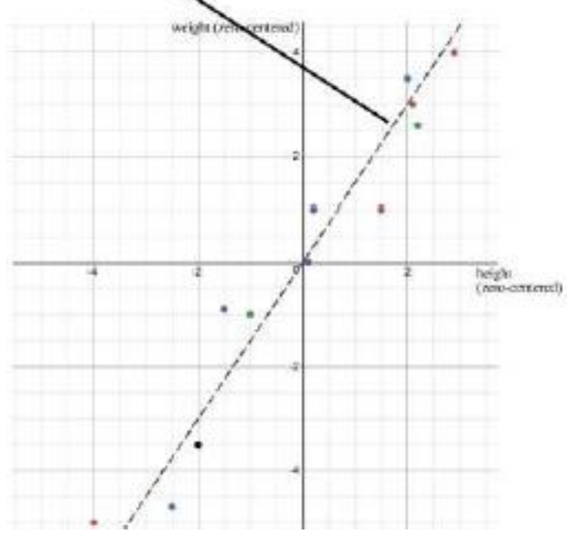
Insight

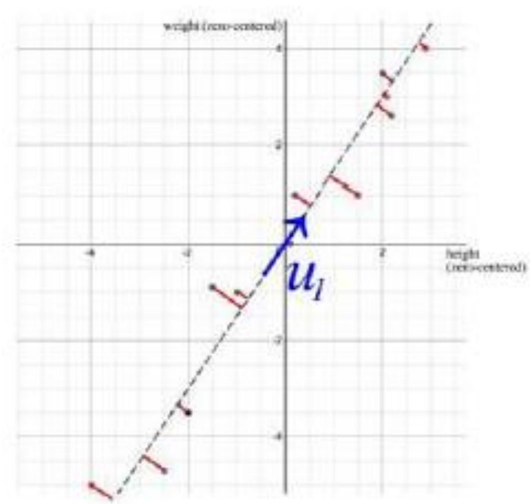
more significant — less significant

$$A = U S V^T = \sigma_1 \underbrace{u_1}_{m \times 1} \underbrace{v_1^T}_{1 \times n} + \ldots + \underbrace{\sigma_r u_r v_r^T}_{m \times n}$$

A: m × n

$$S = \sigma_1 u_1 v_1^T + \cancel{\sigma_2 u_2 v_2^T}$$



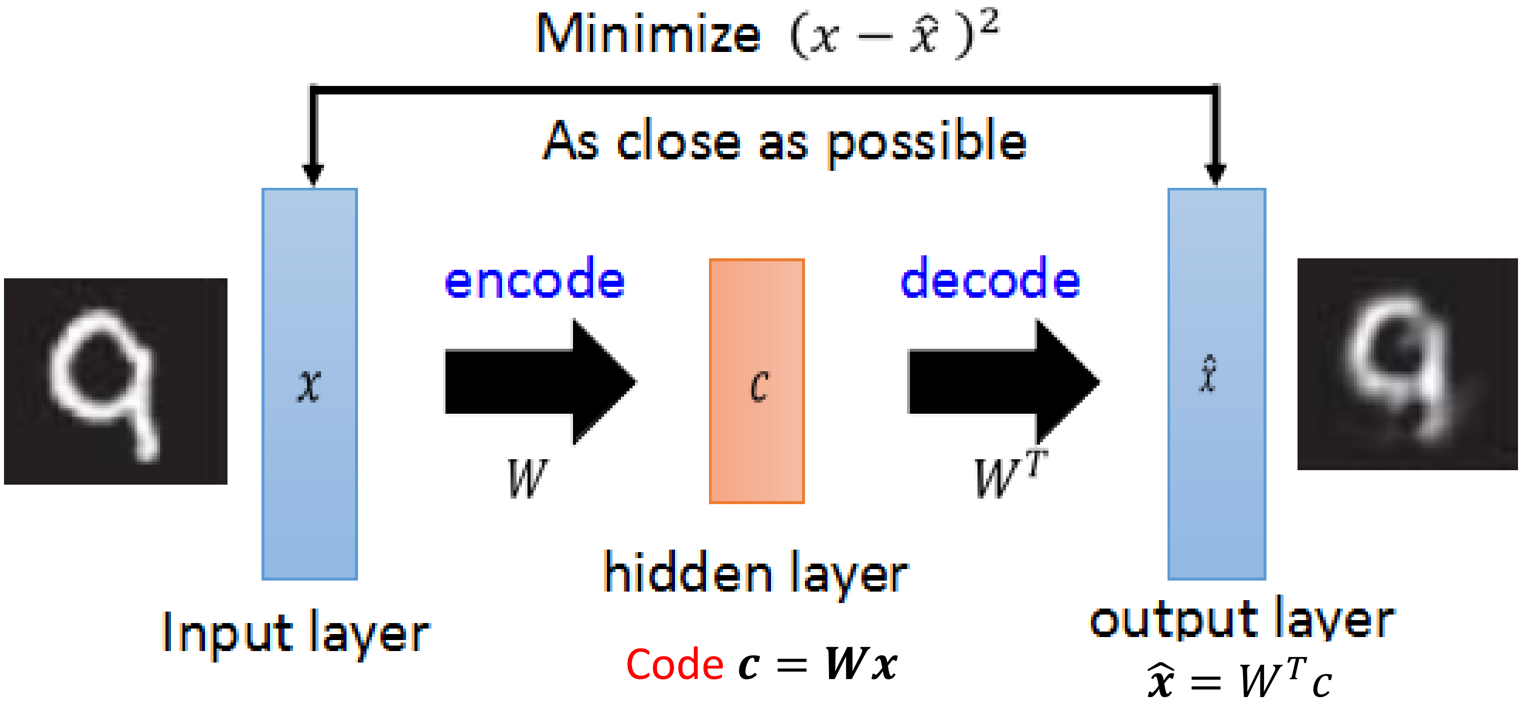


Autoencoder revisit

Auto-Encoder

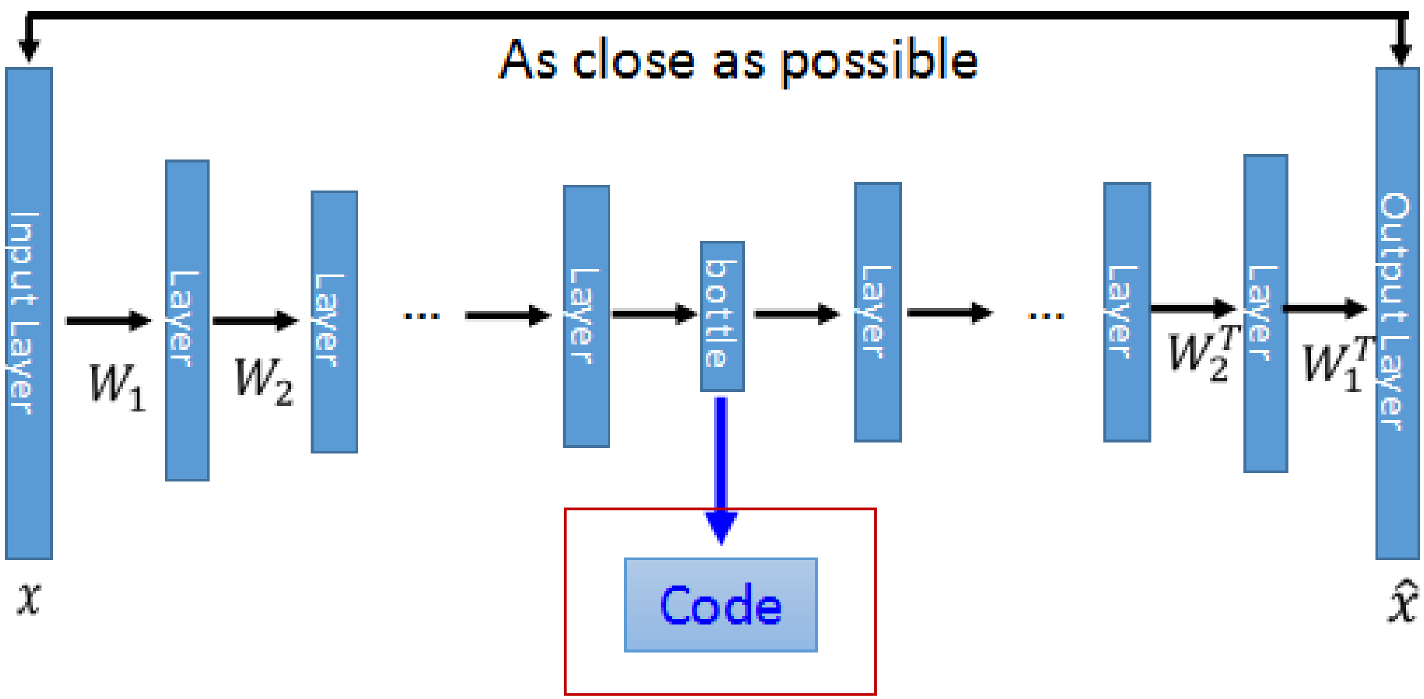
- Auto-Encoder can be used for dimensionality reduction。
- An auto-encoder contains an encoder and a decoder
- The encoder plays the role of coding the original data
  - Data A $\longrightarrow$ Codes
- The decoder plays the role of decoding the “code”
  - Codes $\longrightarrow$ Data B
- Good Auto-Encoder:
  - Data B is very close to Data A

Linear Auto-encoder



Deep Auto-Encoder

- Of course, the Auto-Encoder can be very deep
  - Not necessary to be symmetric



Dimension Reduced

Let’s get our hands dirty!

Data Analysis

Let's do some basic data analysis using our WHO data.

Hide

```
WHO$Under15
```

```
  [1] 47.42 21.33 27.42 15.20 47.58 25.96 24.42 20.34 18.95 14.51 22.25 21.62 20.16 30.57 18.99 15.10 16.88 34.4
0 42.95 28.53
 [21] 35.23 16.35 33.75 24.56 25.75 13.53 45.66 44.20 31.23 43.08 16.37 30.17 40.07 48.52 21.38 17.95 28.03 42.1
7 42.37 30.61
 [41] 23.94 41.48 14.98 16.58 17.16 14.56 21.98 45.11 17.66 33.72 25.96 30.53 30.29 31.25 30.62 38.95 43.10 15.6
9 43.29 28.88
 [61] 16.42 18.26 38.49 45.90 17.62 13.17 38.59 14.60 26.96 40.80 42.46 41.55 36.77 35.35 35.72 14.62 20.71 29.4
3 29.27 23.68
 [81] 40.51 21.54 27.53 14.04 27.78 13.12 34.13 25.46 42.37 30.10 24.90 30.21 35.61 14.57 21.64 36.75 43.06 29.4
5 15.13 17.46
[101] 42.72 45.44 26.65 29.03 47.14 14.98 30.10 40.22 20.17 39.02 35.81 18.26 27.05 19.01 27.85 45.38 25.28 36.5
9 30.10 35.58
[121] 17.21 20.26 33.37 49.99 44.23 30.61 18.64 24.19 34.31 30.10 28.65 38.37 32.78 29.18 34.53 14.91 14.92 13.2
8 15.25 16.52
[141] 15.05 15.45 43.56 25.96 24.31 25.70 37.88 14.04 41.60 29.69 43.54 16.45 21.95 41.74 16.48 15.00 14.16 40.3
7 47.35 29.53
[161] 42.28 15.20 25.15 41.48 27.83 38.05 16.71 14.79 35.35 35.75 18.47 16.89 46.33 41.89 37.33 20.73 23.22 26.0
0 28.65 30.61
[181] 40.54 14.18 14.41 17.54 44.85 19.63 22.05 28.90 37.37 28.84 22.87 40.72 46.73 40.24
```

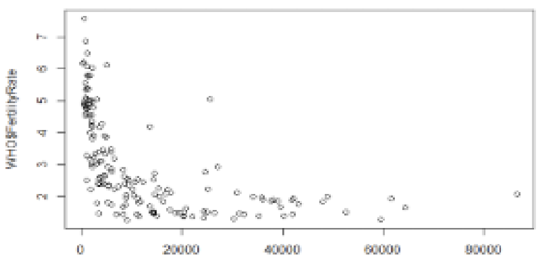
```
WHO$Country[which.min(WHO$Under15)]
```

```
[1] Japan
194 Levels: Afghanistan Albania Algeria Andorra Angola Antigua and Barbuda Argentina Armenia Australia Austria
... Zimbabwe
```

Let's create some plots for exploratory data analysis (EDA). First, Let's create a basic scatterplot of GNI versus FertilityRate.

Hide

```
plot(WHO$GNI, WHO$FertilityRate)
```



EM Method

Problem

- Given the training dataset

$$D = \{x_i\}_{i=1,2,\ldots,N}$$

Let the machine learn the data underlying patterns

- In last lecture, we have learned Clustering
- In EM, we assume latent variables

$$z \rightarrow \boldsymbol{x}$$

- We wish to fit the parameters of a model $p(\boldsymbol{x}, z)$ to the data, where the log-likelihood is

$$\begin{aligned} l(\theta) &= \sum_{i=1}^{N} \log p(x;\theta) \\ &= \sum_{i=1}^{N} \log \sum_{z} p(x,z;\theta) \end{aligned}$$

Method

- Explicitly find the maximum likelihood estimation (MLE) is hard

$$\theta^* = \arg\max_\theta \sum_{i=1}^{N} \log \sum_z p(x^{(i)}, z^{(i)}; \theta)$$

- But given $z^{(i)}$ observed, the MLE is easy

$$\theta^* = \arg\max_\theta \sum_{i=1}^{N} \log p(x^{(i)} | z^{(i)}; \theta)$$

- EM methods give an efficient solution for MLE, by iteratively doing
  - E-step: construct a (good) lower-bound of log-likelihood
  - M-step: optimize that lower-bound

How to construct a good lower-bound?

Lower Bound

- For each instance i, let q_i be some distribution of $z^{(i)}$

$$\sum_z q_i(z) = 1, \quad q_i(z) \geq 0$$

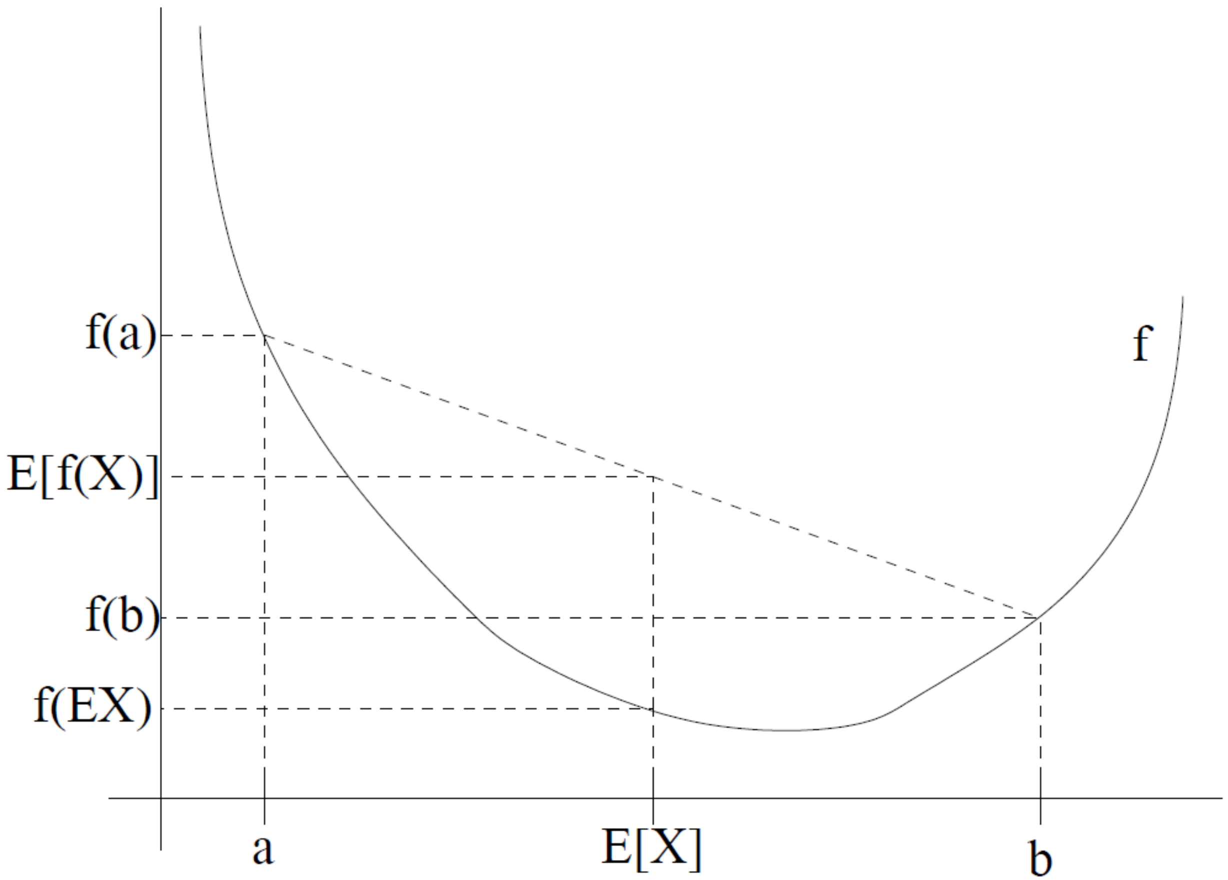
- Thus the data log-likelihood

$$\begin{aligned} l(\theta) = \sum_{i=1}^N \log p(x^{(i)};\theta) &= \sum_{i=1}^N \log \sum_{z^{(i)}} p(x^{(i)}, z^{(i)};\theta) \\ &= \sum_{i=1}^N \log \sum_{z^{(i)}} q_i(z^{(i)}) \frac{p(x^{(i)}, z^{(i)};\theta)}{q_i(z^{(i)})} \\ &\geq \sum_{i=1}^N \sum_{z^{(i)}} q_i(z^{(i)}) \log \frac{p(x^{(i)}, z^{(i)};\theta)}{q_i(z^{(i)})} \end{aligned}$$

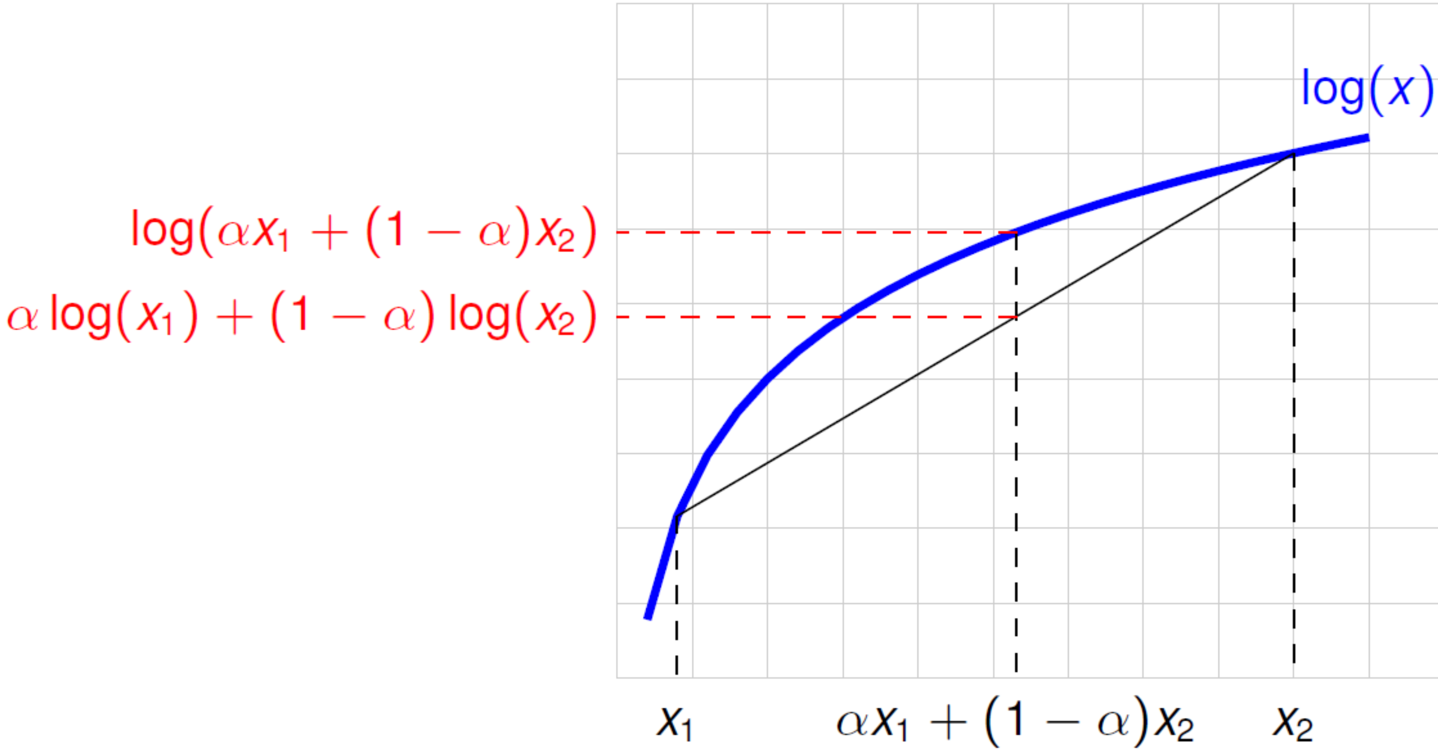
Lower bound of $l(\vartheta)$

Jensen's inequality
-log(x) is a convex function

Jensen’s Inequality



Jensen's Inequality



Lower bound

$$l(\theta) = \sum_{i=1}^{N} \log p(x^{(i)}; \theta) \geq \sum_{i=1}^{N} \sum_{z^{(i)}} q_i(z^{(i)}) \log \frac{p(x^{(i)}, z^{(i)}; \theta)}{q_i(z^{(i)})}$$

- Then what $q_i(z)$ should we choose?
- The posterior distribution

$$q_i(z^{(i)}) = \frac{p(x^{(i)}, z^{(i)}; \theta)}{\sum_z p(x^{(i)}, z; \theta)} = \frac{p(x^{(i)}, z^{(i)}; \theta)}{p(x^{(i)}; \theta)} = p(z^{(i)} | x^{(i)}; \theta)$$

General EM Methods

- Repeat until convergence: {
  - (E-step) For each *i*, set

$$q_i(z^{(i)}) = p(z^{(i)}|x^{(i)};\theta)$$

  - (M-step) Update the parameters

$$\theta = \arg\max_{\theta} \sum_{i=1}^{N} \sum_{z^{(i)}} q_i(z^{(i)}) \log \frac{p(x^{(i)}, z^{(i)}; \theta)}{q_i(z^{(i)})}$$

}

Lecture 14 Wrap-up

- ✓ Dimension Reduction
  - ✓ Motivation
  - ✓ PCA
  - ✓ SVD
  - ✓ Autoencoder revisit
  - ✓ PCA lab
- ✓ EM method

Next Lecture

- Supervised learning
  - Linear regression
  - Logistic regression
  - SVM and kernel
  - Tree models
- Unsupervised learning
  - Clustering
  - PCA (Dimension Reduction)
  - EM
- Deep learning
  - Neural networks
  - Convolutional NN
  - Recurrent NN
- Reinforcement learning
  - MDP
  - ADP
  - Deep Q-Network

Questions?

Shan Wang (王杉)
https://wangshan731.github.io/