Signal(1D)

- Collecting
  - Continuous function of real indepedent variables

  - Practical situation: Collecting the signals in specific time intervals

  - Signals from real world supposed to be continuous and smooth, but limitations of sensor devices can only enable us to collect data at certain time intervals
- Signal types
  - Periodic | Aperiodic

- A signal is periodic if there exists a positive constant T 0 such that

$$f(t+T_0) = f(t) \quad \forall t$$

- Even | Odd

$f(t) = f(-t)$ $\qquad$ $f(t) = -f(-t)$

- Superposition, decomposition

- When dealing with linear systems, signals can only be combined by scaling and adding, no signal signal multiplication
- Synthesis: The process of combining signals through scaling and addition
- The beauty of decomposition : Instead of trying to understand how complicated signals behave as a whole, we study the individual components, which are simpler signals

- Peaks: useful topological features of a time series
  - Indicate significant events.
  - In network distribution data, indicate sudden high demands
  - In server utilization workload, indicate sharp increase in workload
  - Troughs: considered inverted peaks, equally important in many applications
  - Noise in data creates tremendous amount of false positive
  - Some peaks are not result of noise, but still not relevant/desirable in analysis
- Anomaly detection

  - Finding peaks
    - Maxima / Minima

      The maxima and minima are points where

$$\frac{dy}{dx} = f'(x) = 0$$

      - Finite difference
        - Forward difference

$$f'(x) = \frac{f(x + \Delta x) - f(x)}{\Delta x}$$

        - Backward difference

$$f'(x) = \frac{f(x) - f(x - \Delta x)}{\Delta x}$$

        - Central difference

$$f'(x) = \frac{f(x + \Delta x) - f(x - \Delta x)}{2\Delta x}$$

        Delta x is the sampling period
    - Prominence

- measures how much a peak stands out from the surrounding baseline of the signal.
  - 1. Extend a horizontal line from the current peak to the left and right until the line either reaches the window border or intersects the signal again at the slope of a higher peak. An intersection with a peak of the same height is ignored.
  - 2. On each side find the minimal signal value within the interval defined above. These points are the peak's bases.
  - 3. The higher one of the two bases marks the peak's lowest contour line. The prominence can then be calculated as the vertical difference between the peaks height itself and its lowest contour line.
- Width
  - By default, the width of a peak is defined from the position half of of the prominence
- After peaks detected, analysis of peaks involving
  - identifying periodicity of peaks
  - forecasting the time of occurrence and value of next peak
  - identifying dependencies among of peaks of two or more time series
- Baseline correction

- Problem:
  - Sometimes baselines of signals can be unstable; can lead to wrong deduction by algorithms

- The cheaper your sensor, the more you will encounter this problem!
- Drift in sensor values are common phenomenon, due to changes in environment factors such as temperature and humidity

- Correcting baselines can solve this problem and enable easier peak and trough detection, hence promoting easier pattern comparison
- Basic idea

Baseline correction
Basic idea

Derivation steps (NOT important!)

- Assume **y** is the original signal, **z** as the other signal which has this two properties: smooth, faithful to **y**

$$\mathbf{y} = [y_1, y_2, \ldots, y_i, \ldots y_L]$$
$$\mathbf{z} = [z_1, z_2, \ldots, z_i, \ldots z_L]$$

- The baseline can be estimated by minimizing the penalized least squared function

First term here measures the fit to the data; second term is a penalty on non-smooth behavior of z; parameter lambda tunes the balance between the two terms

$$S = \sum_i w_i (y_i - z_i)^2 + \lambda \sum_i (\Delta^2 z_i)^2$$

- where

$$\Delta^2 z_i = (z_i - z_{i-1}) - (z_{i-1} - z_{i-2}) = z_i - 2z_{i-1} + z_{i-2}$$

Baseline correction
Basic idea

- The minimization problems leads to the below system of equations

$$(W + \lambda D'D)z = Wy$$

yc is the parameter of interest but easy to calculate z or yc?

Usually y is known and z is not. Need to find z here first in order to find yc!

What's the solution?

- The baseline corrected signal is then

$$y_c = y - z$$

$$W = \begin{bmatrix} w_1 & 0 & 0 & \cdots & 0 \\ 0 & w_2 & 0 & \cdots & 0 \\ 0 & 0 & w_i & \cdots & 0 \\ \vdots & \vdots & 0 & \ddots & 0 \\ 0 & 0 & 0 & 0 & w_L \end{bmatrix} \qquad D = \begin{array}{c} L\text{-2 columns} \\ \begin{bmatrix} 1 & & & \\ -2 & 1 & & \\ 1 & -2 & \ddots & \\ & 1 & \ddots & 1 \\ & & \ddots & -2 \\ & & & 1 \end{bmatrix} \end{array} \quad L \text{ rows}$$

Weights **Diagonal sparse matrix**

- Similarity between signals
  - two signals by calculating the distance between the i-th point on one signal and the i-th point on another signal

- Consider two distinct signals

$$\mathbf{x} = [x_1, x_2, \ldots, x_i, \ldots x_m]$$
$$\mathbf{y} = [y_1, y_2, \ldots, y_j, \ldots y_n]$$

- The distance between the two signals is defined as

$$d_s(\mathbf{x}, \mathbf{y})$$

- Euclidean distance between two signals:

$$d_s(\mathbf{x}, \mathbf{y}) = \sqrt{\sum_i (x_i - y_i)^2}$$

- **Problem: two signals must be of same length!**

- Simple concept, but could not capture the similarity in shape

- Elastic alignment between points of two signals produces a better, more intuitive similarity measure

- Allow similar shapes to match even if they are out of phase

- Dynamic Time Warping:DTW

- Brief
  - DTW calculates an optimal match between two given sequences

- Sequences are warped along time dimension to determine similarity independent of variations in time
- DTW produces warping path, which enables alignment between two signals

- Pipeline
  - 1. constructing matrix(create warping path)

Dynamic Time Warping
Overview of algorithm

n x m matrix covers the window boundaries

orange curve = correct wrapping path

- Start by constructing n x m matrix D, in which

$$D_{i,j} = d_s(y_i, x_j)$$

- where

$$d_s(y_i, x_j) = (y_i - x_j)^2$$

- Create a warping path w that maps points between x and y, the path w must satisfy the following rules:
  1. Boundary conditions
  2. Monotonicity
  3. Continuity

    - 1st Rule:start point from bottom left and end point at top right
    - 2nd Rule: Cannot move back in direction
    - 3rd Rule: cannot have a break in between! Must be continuous
  - 2. Compute accumulated cost matrix

Cost matrix important to determine wrapping path!

$$A_{i,j} = D_{i,j} + min(A_{i-1,j}, A_{i,j-1}, A_{i-1,j-1})$$

$A_{i,j}$ equals to $D_{i,j}$ plus either $A_{i-1,j-1}$, $A_{i,j-1}$ or $A_{i-1,j}$, whichever has the lowest value

- When i and j equals to 0

$$A_{0,0} = D_{0,0}$$

- When i equals to 0 (first row)

$$A_{0,j} = D_{0,j} + A_{0,j-1}$$

- When j equals to 0 (first column)

$$A_{i,0} = D_{i,0} + A_{i-1,0}$$

Dynamic Time Warping
2. Compute accumulated cost matrix

- When i equals to 0 (first row)

$$A_{0,j} = D_{0,j} + A_{0,j-1}$$

- When j equals to 0 (first column)

$$A_{i,0} = D_{i,0} + A_{i-1,0}$$

$\longrightarrow j$

D

0.25	0.25	1	4	1	1
4	4	0.25	0.25	0.25	6.25
6.25	6.25	1	0	1	9
1	1	0.25	2.25	0.25	2.25
1	1	0.25	2.25	0.25	2.25
1	1	0.25	2.25	0.25	2.25
0.25	0.25	4	9	4	0

$\uparrow i$

A

9.5					
3.25					
2.25					
1.25					
0.25	0.5	4.5	13.5		

Dynamic Time Warping
2. Compute accumulated cost matrix

- Else

$$A_{i,j} = D_{i,j} + min(A_{i-1,j}, A_{i,j-1}, A_{i-1,j-1})$$

$\longrightarrow j$

D

0.25	0.25	1	4	1	1
4	4	0.25	0.25	0.25	6.25
6.25	6.25	1	0	1	9
1	1	0.25	2.25	0.25	2.25
1	1	0.25	2.25	0.25	2.25
1	1	0.25	2.25	0.25	2.25
0.25	0.25	4	9	4	0

$\uparrow i$

A

13.75					
13.5					
9.5	9.5	2.25	1.25	2.25	
3.25	3.25	1.25	3.25	3.25	5.5
2.25	2.25	1	3	3.25	5.5
1.25	1.25	0.75	3	3.25	5.5
0.25	0.5	4.5	13.5	17.5	17.5

- 3. Search the optimal path

- Wrapping process C ompare 3 neighboring squares and find the minimum cost square

- Comparison

	Pros	Cons
Euclidean/Manhattan distance	Easy to implement; straightforward	Only works if the signals to be compared are of same length and preferably similar shape
DTW	Not necessary for signals to be of same length and shape; Computes faster than LCSS	Heavy computational burden (worse than Euclidean/Manhattan distance method); Unable to work if one of the signals is a partial type
Longest common subsequence (LCSS)	Not necessary for signals to be of same length and shape; Allows for the use of partial and noisy signals	Heavy computational burden; Computes slower than DTW
Developed Longest Common Subsequence (DLCSS)	Not necessary for signals to be of same length and shape; Allows for the use of partial and noisy signals; Most accurate among the other methods	Heavy computational burden; Computes slower than DTW and LCSS

- Signal decomposition
  - Image signal decomposition: Checkerboard basis.

Input image

A set of basis (checkerboard)

Note: The images are NOT visualized in the actual scale.

  - Images have some slow/fast varying regions. For example, a first checkerboard picture with (slow varying) constant color. A second checkerboard picture that has fast changes.

- Sound: Sinusoids basis.

Periodic functions

General form of sine and cosine functions:

$\lvert A \rvert$	amplitude
$\frac{2\pi}{\lvert \alpha \rvert}$	period
b	phase shift

$$y(t) = A\sin(\alpha t + b) \qquad y(t) = A\cos(\alpha t + b)$$

Example: $A = 1, b = 0, \alpha = 1 \quad \text{period} = 2\pi$

- They are orthogonal.
- They can represent rounded shapes nicely.

- Idea: Decompose the input signal from the time domain into a set of basis (that are sinusoid functions) to obtain the decomposed signal in the frequency domain.

- Fourier Analysis

Signal (Fourier domain) | Apply all signal values | Signal (time domain) | Basis (sinusoid functions)

Forward: $F(u) = \sum_{x=0}^{N-1} f(x) e^{\frac{-i2\pi ux}{N}}$, where $u = 0,1,\cdots,N-1$

Inverse: $f(x) = \frac{1}{N}\sum_{u=0}^{N-1} F(u) e^{\frac{i2\pi ux}{N}}$, where $x = 0,1,\cdots,N-1$

Note: $e^{ix} = \cos x + i \sin x$; $e^{i\pi} = \cos\pi + i\sin\pi = -1$, Reference: https://en.wikipedia.org/wiki/Euler%27s_identity

Example

- Signal: $f(x) = [2,3,4,4]$
- Fourier coefficients: $F(u) = [13, (-2+i), -1, (-2-i)]$, where i is the imaginary unit

$$F(0) = \sum_{x=0}^{3} f(x) e^{\frac{-i2\pi 0x}{4}} = 2+3+4+4 = 13$$

$$F(1) = \sum_{x=0}^{3} f(x) e^{\frac{-i2\pi x}{4}} = 2e^0 + 3e^{-i\pi/2} + 4e^{-i\pi} + 4e^{-i3\pi/2} = -2+i$$

$$F(2) = \sum_{x=0}^{3} f(x) e^{\frac{-i4\pi x}{4}} = 2e^0 + 3e^{-i\pi} + 4e^{-i2\pi} + 4e^{-i3\pi} = -1$$

$$F(3) = \sum_{x=0}^{3} f(x) e^{\frac{-i6\pi x}{4}} = 2e^0 + 3e^{-i3\pi/2} + 4e^{-i3\pi} + 4e^{-i9\pi/2} = -2-i$$

What is the Fourier Transform? A visual introduction
A 20-minute video tutorial on
https://www.youtube.com/watch?v=spUNpyF58BY
4.4 millions views since January 2018

- Challenge: Provides good localization in the frequency domain but poor localization in the time domain. Has knowledge of what frequencies exist , but no information about where these frequencies are located in time

Example 1: $f(t) = \cos(2\pi \cdot 5 \cdot t) + \cos(2\pi \cdot 25 \cdot t)$

Example 2: $f(t) = \begin{cases} \cos(2\pi \cdot 5 \cdot t) & 0 \le t < 1 \\ \cos(2\pi \cdot 25 \cdot t) & 1 \le t < 2 \end{cases}$

- Short time Fourier transform

  - Idea: Segment the signal into narrow time intervals (i.e., narrow enough to be considered stationary) and take the Fourier transform of each segment.
    - Choose a window function of finite length
    - Place the window on start of the signal
    - Truncate the signal using this window
    - Compute Fourier transform of the truncated signal
    - Incrementally slide the window to the right

        - Repeat until window reaches the end of the signal
- Wavelet

    - Brief
        - Varying frequency
        - Limited duration
    - Decomposition
        - Decompose signal by averaging and differencing the input signal to get approximation coefficients and detail coefficients, followed by downsampling

Result: Wavelet coefficients
[6 2 1 −1]

        - other choices

- 2D wavelet transformation

  - AA: The upper left quadrant is filtered by the averaging filter along the rows and then filtered along the corresponding columns with the averaging filter. It represents the approximation coefficients of the original at half the resolution.
  - DA/AD: The lower left and the upper right blocks are filtered along the rows and columns with averaging filter and differencing filter, alternatively. The AD block contains vertical edges. In contrast, the HL blocks shows horizontal edges.
  - DD: The lower right quadrant is derived analogously to the upper left quadrant but with the use of the differencing filter, where we find edges of the original image in diagonal direction.
- Example:
  - Reconstruction

    Given the wavelet coefficients (obtained in previous example)
    $[6 \quad 2 \quad 1 \quad -1]$

    $$[6] \xrightarrow{2} [8 \quad 4] \xrightarrow{[1\ -1]} [9 \quad 7 \quad 3 \quad 5]$$

    (6+2) (6-2) (8+1) (8-1) (4+(-1)) (4-(-1))

    - The original signal can be reconstructed by adding or subtracting the detail coefficients from the approximation coefficients.
  - Denoising

- Signal
- Wavelet decomposition
- Manipulate wavelet coefficients
- Wavelet reconstruction
- Reconstructed signal

- Feature extraction
  - Perform wavelet transformation on signal
  - Extract statistical features
    - Auto-regressive model coefficient values
    - Statistical features like variance, mean, median, etc.
  - Perform signal classification or other machine learning tasks

- Statistical signal analysis
  - Learning: For a given set of observations $\boldsymbol{X} = \boldsymbol{X}1, \boldsymbol{X}2,, \cdots, \boldsymbol{X}N$ from some random variables, learn a model $f(\boldsymbol{X}|\boldsymbol{\theta})$ (defined by the parameter $\boldsymbol{\theta}$) to best describe the data.
  - Prediction: Assume that we have two classes of objects as $\omega\omega1$ and $\omega\omega2$, and we already have learned models $f(\boldsymbol{X}|\boldsymbol{\theta 1})$ and $f(\boldsymbol{X}|\boldsymbol{\theta 2})$ for $\omega1$ and $\omega2$, respectively. For a new observation $\boldsymbol{X}n$, predict which class it belongs to, that is $\boldsymbol{f}$ $(\omega1, \omega2|\boldsymbol{X}n, \boldsymbol{\theta 1}, \boldsymbol{\theta 2})$
  - Signal classification
    - Given a signal x, two categories of data, say ω_1, ω_2, does the signal x belong to ω_1 or ω_2?
    - That means, we need to evaluate $P(\omega_1|x)$ and $P(\omega_2|x)$ and make decision (next slide).
    - The class posterior probability is

Likelihood Prior

$$P(\omega_i|x) = \frac{P(x|\omega_i)P(\omega_i)}{P(x)}$$

Evidence

- Class priors $P(\omega i)$
  - How much of each class? $P\ (\omega i\ ) \approx Ni/N$
- Class likelihood $P(x|\omega i)$
  - Requires that we have a model for each ωi, for example, ωi can be modelled as Gaussian distribution
- Evidence
  - $P\ (x) = P(x|\omega 1)\ P\ (\omega 1) + P(x|\omega 2\ )P(\omega 2)$
- Classification rule
  - If $P(\omega 1|x) \geq P(\omega 2|x)$,then x is classified as $\omega 1$
  - If $P(\omega 1|x) < P(\omega 2|x)$,then x is classified as $\omega 2$
- Example
  - discrete data

Given: $\boldsymbol{x}$=(Outlook=*Sunny*, Temperature=*Cool*, Humidity=*High*, Wind=*Strong*)
Predict: PlayTennis Yes or No?

PlayTennis: training examples

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

Given: $\boldsymbol{x}$=(Outlook=*Sunny*, Temperature=*Cool*, Humidity=*High*, Wind=*Strong*)
Predict: PlayTennis Yes or No?

Bayesian Rule	
P(*Yes*|$\boldsymbol{x}$)	0.0053
[P(*Sunny*|Yes)P(*Cool*| *Yes*)P(*High*|Yes)P(*Strong*| *Yes*)]P(Play=*Yes*)	
P(*No*|$\boldsymbol{x}$)	0.0206
[P(*Sunny*|No) P(*Cool*|No)P(*High*|*No*)P(*Strong*|*No*)]P(Play=*No*)	

Decision: Given the fact $P(Yes|\boldsymbol{x}) < P(No|\boldsymbol{x})$, we decide $\boldsymbol{x}$ to be *No*.

P(Outlook=*Sunny*|Play=*Yes*) = 2/9
P(Temperature=*Cool*|Play=*Yes*) = 3/9
P(Huminity=*High*|Play=*Yes*) = 3/9
P(Wind=*Strong*|Play=*Yes*) = 3/9
P(Play=*Yes*) = 9/14

P(Outlook=*Sunny*|Play=*No*) = 3/5
P(Temperature=*Cool*|Play==*No*) = 1/5
P(Huminity=*High*|Play=*No*) = 4/5
P(Wind=*Strong*|Play=*No*) = 3/5
P(Play=*No*) = 5/14

Outlook	Play=*Yes*	Play=*No*
Sunny	2/9	3/5
Overcast	4/9	0/5
Rain	3/9	2/5

Temperature	Play=*Yes*	Play=*No*
Hot	2/9	2/5
Mild	4/9	2/5
Cool	3/9	1/5

Humidity	Play=*Yes*	Play=No
High	3/9	4/5
Normal	6/9	1/5

Wind	Play=*Yes*	Play=*No*
Strong	3/9	3/5
Weak	6/9	2/5

Total: $P(Play = Yes) = 9/14, P(Play = No) = 5/14$

- continuous data

For the training data

- Calculate the prior probability by counting the number of data in each class. $P(Window\ Glass)$ and $P(Not\ Window\ Glass)$
- Build the likelihood probability distribution, for each class, assume each attribute (column) follows Gaussian distribution, calculate its mean and variance.

For the test data

- Apply the same formula. Calculate the likelihood probability using the built Gaussian functions, $P(Refractive\ index = 1.71720|Window\ Glass)$ and $P(Refractive\ index = 1.71720|Not\ Window\ Glass)$

Training data

No.	Refractive index	Sodium	Magnesium	Aluminum	Silicon	Potassium	Calcium	Barium	Iron	Class
1	1.51824	12.87	3.48	1.29	72.95	0.60	8.43	0.00	0.00	Window Glass
2	1.51832	13.33	3.34	1.54	72.14	0.56	8.99	0.00	0.00	Window Glass
3	1.51747	12.84	3.50	1.14	73.27	0.56	8.55	0.00	0.00	Window Glass
4	1.51775	12.85	3.48	1.23	72.97	0.61	8.56	0.09	0.22	Window Glass
5	1.51768	12.65	3.56	1.30	73.08	0.61	8.69	0.00	0.14	Window Glass
6	1.51769	12.45	2.71	1.29	73.70	0.56	9.06	0.00	0.24	Window Glass
7	1.51892	13.46	3.83	1.26	72.55	0.57	8.21	0.00	0.14	Window Glass
8	1.51727	14.70	0.00	2.34	73.28	0.00	8.95	0.66	0.00	Not Window Glass
9	1.51545	14.14	0.00	2.68	73.39	0.08	9.07	0.61	0.05	Not Window Glass
10	1.51994	13.27	0.00	1.76	73.03	0.47	11.32	0.00	0.00	Not Window Glass

Test data

100	1.51720	13.38	3.50	1.15	72.85	0.50	8.43	0.00	0.00

- Learning distribution
  - easy data

Bernoulli distribution

$$p(\mathbf{x}|\boldsymbol{\mu}) = \prod_i \mu_i^{x_i}(1-\mu_i)^{x_i}$$

Gaussian distribution

$$p(\mathbf{x}|\boldsymbol{\mu},\Sigma) = \frac{1}{\sqrt{(2\pi)^N|\Sigma|}} exp\left\{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})\Sigma^{-1}(\mathbf{x}-\boldsymbol{\mu})\right\}$$

    - Problem:Given a collection of examples,estimate its single distribution model
    - Solution:
      - Assign a model to the distribution
      - Learn parameters of model from data
  - complicated data

    - We need to perform data clustering(data labelling) and distribution estimation together
    - example

Example: **6 3 1 5 4 1 2 4 ...**

- A person shoots a loaded dice repeatedly
- You observe the series of outcomes
- You can form a good idea of how the dice is loaded, by finding what the probabilities of the various numbers are for dice
- $\text{Probability}(number) = \frac{\text{Count}(number)}{\text{Sum}(rolls)}$

- Additional rules
  - Two dice (say, **yellow dice** and **blue dice**) are available.
  - The dice are differently loaded for the two of them.
  - We observe the series of outcome.
  - There is a "caller" who randomly calls out the outcomes.
  - At any time, you do not know which of the two dice you are calling out,

X	1	2	3	4	5	6
Initially, we randomly guess						
$P(X\|\text{Blue})$	0.3	0.3	0.1	0.1	0.1	0.1
$P(X\|\text{Yellow})$	0.4	0.05	0.05	0.05	0.05	0.4
Currently, we have updated results after one iteration of update						
$P(X\|\text{Blue})$	0.122	0.322	0.125	0.250	0.125	0.056
$P(X\|\text{Yellow})$	0.234	0.077	0.090	0.181	0.090	0.328

	Previous random guess	Updated results
$P(Z = \text{Blue})$	0.5	0.59
$P(Z = \text{Yellow})$	0.5	0.41

The *Expectation Maximization* (EM) algorithm will continue until the stopping criterion (e.g., # of iterations, etc).

- Step 1: Initialization
  - We (guess) obtain an initial estimate for the probability distribution of the two sets of dice:

X	1	2	3	4	5	6
$P(X\|\text{Blue})$	0.3	0.3	0.1	0.1	0.1	0.1
$P(X\|\text{Yellow})$	0.4	0.05	0.05	0.05	0.05	0.4

  - We (guess) obtain an initial estimate for the probability with which the caller calls out the two shooters

$P(Z = \text{Blue})$	0.5
$P(Z = \text{Yellow})$	0.5

- Step 2: Estimate hidden labels

- Every observed roll of the dice contributes to both "Yellow" and "Blue"

6 4 5 1 2 3 4 5 2 2 1 4 3 4 6 2 1 6

Probability to be Yellow	Probability to be Blue
6 (0.8), 4 (0.33), 5 (0.33), 1 (0.57), 2 (0.14), 3 (0.33), 4 (0.33), 5 (0.33), 2 (0.14), 2 (0.14), 1 (0.57), 4 (0.33), 3 (0.33), 4 (0.33), 6 (0.8), 2 (0.14), 1 (0.57), 6 (0.8)	6 (0.2), 4 (0.67), 5 (0.67), 1 (0.43), 2 (0.86), 3 (0.67), 4 (0.67), 5 (0.67), 2 (0.86), 2 (0.86), 1 (0.43), 4 (0.67), 3 (0.67), 4 (0.67), 6 (0.2), 2 (0.86), 1 (0.43), 6 (0.2)

Recall that we randomly guess in Step 1

$P(Z = \text{Blue}) = P(Z = \text{Yellow}) = 0.5$
$P(6|\text{Blue}) = 0.1, P(6|\text{Yellow}) = 0.4$

For roll number 6, we estimate its label

$P(\text{Yellow}|X = 6) = P(X = 6|Z = \text{Yellow})P(Z = \text{Yellow}) = 0.4 \times 0.5 = 0.2$
$P(\text{Blue}|X = 6) = P(X = 6|Z = \text{Blue})P(Z = \text{Blue}) = 0.1 \times 0.5 = 0.05$
After normalization, we have $P(\text{Yellow}|X = 6) = 0.8, P(\text{Blue}|X = 6) = 0.2$

- Every observed roll of the dice contributes to both "Yellow" and "Blue"
- Total count for "Yellow" is the sum of all the posterior probabilities in the Yellow column: 7.31
- Total count for "Blue" is the sum of all the posterior probabilities in the Blue column: 10.69

Called	$P(\text{Yellow}\|X)$	$P(\text{Blue}\|X)$
6	.8	.2
4	.33	.67
5	.33	.67
1	.57	.43
2	.14	.86
3	.33	.67
4	.33	.67
5	.33	.67
2	.14	.86
2	.14	.86
1	.57	.43
4	.33	.67
3	.33	.67
4	.33	.67
6	.8	.2
2	.14	.86
1	.57	.43
6	.8	.2
	SUM=7.31	SUM=10.69

- Step 3: Update $P(X|\text{Yellow})$

- Total count for "Yellow": 7.31
  - Total count for 1: 1.71
  - Total count for 2: 0.56
  - Total count for 3: 0.66
  - Total count for 4: 1.32
  - Total count for 5: 0.66
  - Total count for 6: 2.4
- Updated probability of Yellow dice:
  - $P(1 \mid \text{Yellow}) = 1.71/7.31 = 0.234$
  - $P(2 \mid \text{Yellow}) = 0.56/7.31 = 0.077$
  - $P(3 \mid \text{Yellow}) = 0.66/7.31 = 0.090$
  - $P(4 \mid \text{Yellow}) = 1.32/7.31 = 0.181$
  - $P(5 \mid \text{Yellow}) = 0.66/7.31 = 0.090$
  - $P(6 \mid \text{Yellow}) = 2.40/7.31 = 0.328$

Called	$P(\text{Yellow}\|X)$	$P(\text{Blue}\|X)$
6	.8	.2
4	.33	.67
5	.33	.67
1	.57	.43
2	.14	.86
3	.33	.67
4	.33	.67
5	.33	.67
2	.14	.86
2	.14	.86
1	.57	.43
4	.33	.67
3	.33	.67
4	.33	.67
6	.8	.2
2	.14	.86
1	.57	.43
6	.8	.2
	SUM=7.31	SUM=10.69

- Total count for "Blue": 10.69
  - Total count for 1: 1.29
  - Total count for 2: 3.44
  - Total count for 3: 1.34
  - Total count for 4: 2.68
  - Total count for 5: 1.34
  - Total count for 6: 0.6
- Updated probability of Blue dice:
  - $P(1 \mid \text{Blue}) = 1.29/11.69 = 0.122$
  - $P(2 \mid \text{Blue}) = 0.56/11.69 = 0.322$
  - $P(3 \mid \text{Blue}) = 0.66/11.69 = 0.125$
  - $P(4 \mid \text{Blue}) = 1.32/11.69 = 0.250$
  - $P(5 \mid \text{Blue}) = 0.66/11.69 = 0.125$
  - $P(6 \mid \text{Blue}) = 2.40/11.69 = 0.0563$

Called	$P(\text{Yellow}\|X)$	$P(\text{Blue}\|X)$
6	.8	.2
4	.33	.67
5	.33	.67
1	.57	.43
2	.14	.86
3	.33	.67
4	.33	.67
5	.33	.67
2	.14	.86
2	.14	.86
1	.57	.43
4	.33	.67
3	.33	.67
4	.33	.67
6	.8	.2
2	.14	.86
1	.57	.43
6	.8	.2
	SUM=7.31	SUM=10.69

- Step 4: Update $P(Z$=Blue) and $P(Z$=Yellow)
  - Total count for "Yellow": 7.31
  - Total count for "Blue": 10.69
  - Total instances: $7.31 + 10.69 = 18$
  - We also normalize our estimate for the probability that the caller calls out Yellow or Blue
  - $P(Z = \text{Yellow}) = 7.31/18 = 0.41$
  - $P(Z = \text{Blue}) = 10.69/18 = 0.59$

Called	$P(\text{Yellow}\|X)$	$P(\text{Blue}\|X)$
6	.8	.2
4	.33	.67
5	.33	.67
1	.57	.43
2	.14	.86
3	.33	.67
4	.33	.67
5	.33	.67
2	.14	.86
2	.14	.86
1	.57	.43
4	.33	.67
3	.33	.67
4	.33	.67
6	.8	.2
2	.14	.86
1	.57	.43
6	.8	.2
	SUM=7.31	SUM=10.69

- Signal representation
  - 1.Signal representation can be manually designed (input-agnostic)

  - 2. Signal representation can be adaptively to the input signal

  - 3. Signal representation can be learned from signal dataset

  - Requirements
    - Smoothness

- By “smoothness” we mean that close inputs are mapped to close outputs (representations).
- To fit the data (green stars), the blue model is smoother than the purple model.

- Dimensionality
  - Sometime, we need a representation with a lower dimension than the input dimension, to avoid complicated calculations or having too many configurations. But note that we should not loose too much important data.

  - Sometime, we also need to increase dimensions of data to obtain richer representations.

- Depth and abstraction

  - A good representation expresses high level and abstract features. In order to achieve such representations we can use deep architectures that allow reuse of low level features to potentially get more abstract features at higher layers.

- Basic vectors

Signal	$\mathbf{x} = (1, 2)$
Basis vectors	$\mathbf{v}_1 = (1, 0), \mathbf{v}_2 = (0, 1)$
Signal representation coefficients	$\mathbf{w} = (1, 2)$
Justification	$\mathbf{x} = 1 \times (1, 0) + 2 \times (0, 1)$

  - The basis vector determines how the signal is represented.We can change basis vectors so that we can change signal representation
  - change in a different orthonormal basis

Signal	$\mathbf{x} = (1, 2)$
Basis vectors	$\mathbf{u}_1 = \left(\sqrt{2}/2, \sqrt{2}/2\right), \mathbf{u}_2 = \left(-\sqrt{2}/2, \sqrt{2}/2\right)$
Signal representation coefficients	$\mathbf{w} = \left(3\sqrt{2}/2, \sqrt{2}/2\right)$
Justification	$\mathbf{x} = 3\sqrt{2}/2 \times \left(\sqrt{2}/2, \sqrt{2}/2\right) + \sqrt{2}/2 \times \left(-\sqrt{2}/2, \sqrt{2}/2\right)$

- Decompose signal $\mathbf{x}$

$$w_i = \langle \mathbf{x}, \mathbf{u}_i \rangle = \mathbf{x}^T \mathbf{u}_i = \sum_j x(j) u_i(j)$$

where <·> is the dot product of two vectors

- Reconstruct signal $\mathbf{x}$

$$\mathbf{x} = \sum_i w_i \times \mathbf{u}_i$$

- Checkerboard basis

Signal at standard basis: $\mathbf{x} = \begin{bmatrix} 2 & 1 \\ 6 & 1 \end{bmatrix}$

Question: All the basis we have considered so far are data agnostic. Checkerboards, Complex exponentials, Wavelets.. We use the same bases regardless of the data we analyze. How about data specific bases that consider the underlying data? Is there something better than checkerboards?

$$\begin{bmatrix} 2 & 1 \\ 6 & 1 \end{bmatrix} = 2 \times \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} + 1 \times \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} + 6 \times \begin{bmatrix} 0 & 0 \\ 1 & 0 \end{bmatrix} + 1 \times \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix}$$

New basis Signal at new basis: $\mathbf{x} = \begin{bmatrix} 5 & -2 \\ 2 & -3 \end{bmatrix}$

$$\mathbf{u}_1 = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}/2 \qquad \mathbf{u}_2 = \begin{bmatrix} 1 & 1 \\ -1 & -1 \end{bmatrix}/2$$

$$\mathbf{u}_3 = \begin{bmatrix} -1 & 1 \\ 1 & -1 \end{bmatrix}/2 \qquad \mathbf{u}_4 = \begin{bmatrix} -1 & 1 \\ -1 & 1 \end{bmatrix}/2$$

Recall the formula

$$w_i = \langle \mathbf{x}, \mathbf{u}_i \rangle = \mathbf{x}^T \mathbf{u}_i = \sum_j x(j) u_i(j)$$

where <·> is the dot product of two vectors

$$\mathbf{x} = \sum_i w_i \times \mathbf{u}_i$$

- Energy compaction property

Note: N might not be as same as the dimension of $\mathbf{x}$

How to define better basis for signal representation?

- Given the signal representation as $\mathbf{x} = \sum_{k=1}^{N} w_k \mathbf{u}_k$
- The ideal is $\hat{\mathbf{x}} = w_1 \mathbf{u}_1 + w_2 \mathbf{u}_2 + w_3 \mathbf{u}_3 + \cdots + w_N \mathbf{u}_N$ based on N basis components. Its error is defined as $\text{Error}_N = \|\mathbf{x} - \hat{\mathbf{x}}\|^2$
- If the signal representation is terminated at any point, we should still get most of the information about the data, that means $\text{Error}_N < \text{Error}_{N-1}$

- Assumption: There are a set of N "typical" basis vectors that captures most of all (say, M) input data $\mathbf{x}_i$, where $i = 1, \cdots, M$.
- Approximate every data $\mathbf{x}_i$ as $\hat{\mathbf{x}}_i = w_{i,1} \boldsymbol{u}_1 + w_{i,2} \boldsymbol{u}_2 + \cdots + w_{i,N} \boldsymbol{u}_N$
  - $\mathbf{u}_2$ is used to "correct" errors resulting from using only $\mathbf{u}_1$.
  - $\left\|\mathbf{x}_i - \left(w_{i,1} \boldsymbol{u}_1 + w_{i,2} \boldsymbol{u}_2\right)\right\|^2 < \left\|\mathbf{x}_i - w_{i,1} \boldsymbol{u}_1\right\|^2$
  - $\mathbf{u}_3$ corrects errors remaining after correction with $\mathbf{u}_2$
  - $\left\|\mathbf{x}_i - \left(w_{i,1} \boldsymbol{u}_1 + w_{i,2} \boldsymbol{u}_2 + + w_{i,3} \boldsymbol{u}_3\right)\right\|^2 < \left\|\mathbf{x}_i - \left(w_{i,1} \boldsymbol{u}_1 + w_{i,2} \boldsymbol{u}_2\right)\right\|^2$
  - And so on
- Estimate $\mathbf{u}_1, \mathbf{u}_2, \cdots, \mathbf{u}_N$ to minimize the squared error between the original signal and the reconstructed signal.

- Find first vector

Each "point" represents a signal data (displayed as image for visualization).

Any "basis vector" $\mathbf{u}$ is a vector in this space.

The approximation $\mathbf{u}\mathbf{u}^T\mathbf{x}$ for any signal $\mathbf{x}$ is the *projection* of $\mathbf{x}$ onto $\mathbf{u}$.

The distance between $\mathbf{x}$ and its projection $\mathbf{u}\mathbf{u}^T\mathbf{x}$ is the *projection error.*

Every signal data will suffer error when approximated by its projection on $\mathbf{u}$

The total squared length of all error lines is the *total squared projection error.*

The problem of finding the first basis vector: Find the $\mathbf{u}$ for which the total projection error is minimum!

This "minimum squared error" $\mathbf{u}$ is our "best" first typical basis vector.

Projection of a vector $\mathbf{x}$ on to a vector $\mathbf{u}$, which has unit length ($\|\mathbf{u}\| = 1$), $\hat{\mathbf{x}} = \mathbf{u}\mathbf{u}^T\mathbf{x}$

$$
\begin{aligned}
Error &= \|\mathbf{x} - \hat{\mathbf{x}}\|^2 = \|\mathbf{x} - \mathbf{u}\mathbf{u}^T\mathbf{x}\|^2 \\
&= (\mathbf{x} - \mathbf{u}\mathbf{u}^T\mathbf{x})^T(\mathbf{x} - \mathbf{u}\mathbf{u}^T\mathbf{x}) \\
&= \mathbf{x}^T\mathbf{x} - \mathbf{x}^T\mathbf{u}\mathbf{u}^T\mathbf{x} - \mathbf{x}^T\mathbf{u}\mathbf{u}^T\mathbf{x} + \mathbf{x}^T\mathbf{u}\mathbf{u}^T\mathbf{u}\mathbf{u}^T\mathbf{x} \\
&= \mathbf{x}^T\mathbf{x} - \mathbf{x}^T\mathbf{u}\mathbf{u}^T\mathbf{x}
\end{aligned}
$$

$\boxed{\mathbf{u}^T\mathbf{u}=1}$

- Error for one signal vector $\mathbf{x}$ $Error = \mathbf{x}^T\mathbf{x} - \mathbf{x}^T\mathbf{u}\mathbf{u}^T\mathbf{x}$
- Error for all signal vectors $\mathbf{x}_i$

$$Error = \sum_i (\mathbf{x}_i^T\mathbf{x}_i - \mathbf{x}_i^T\mathbf{u}\mathbf{u}^T\mathbf{x}_i) = \sum_i \mathbf{x}_i^T\mathbf{x}_i - \sum_i \mathbf{x}_i^T\mathbf{u}\mathbf{u}^T\mathbf{x}_i$$

- Goal: Find $\mathbf{u}$ to minimize this error.

$$Error = \mathbf{x}^T\mathbf{x} - \mathbf{x}^T\mathbf{u}\mathbf{u}^T\mathbf{x}$$

Find $\mathbf{u}$ to minimize the error $\sum_i \mathbf{x}_i^T\mathbf{x}_i - \sum_i \mathbf{x}_i^T\mathbf{u}\mathbf{u}^T\mathbf{x}_i$ subject to $\mathbf{u}^T\mathbf{u} = 1$

unit vect

Apply Lagrange multiplier α and set derivative (with respect to $\mathbf{u}$) to be zero

$$C = \sum_i \mathbf{x}_i^T\mathbf{x}_i - \sum_i \mathbf{x}_i^T\mathbf{u}\mathbf{u}^T\mathbf{x}_i + \alpha(\mathbf{u}^T\mathbf{u} - 1)$$

$$-2\sum_i \mathbf{x}_i\mathbf{x}_i^T\mathbf{u} + 2\alpha\mathbf{u} = 0 \Rightarrow \boxed{\left(\sum_i \mathbf{x}_i\mathbf{x}_i^T\right)}\mathbf{u} = \alpha\mathbf{u}$$

- $\mathbf{u}$: eigenvector of matrix $\sum_i \mathbf{x}_i\mathbf{x}_i^T$
- α is eigenvalue

Minimize $C = \sum_i \mathbf{x}_i^T\mathbf{x}_i - \sum_i \mathbf{x}_i^T\mathbf{u}\mathbf{u}^T\mathbf{x}_i$	Recall $\mathbf{x}_i^T\mathbf{u} = \mathbf{u}^T\mathbf{x}_i$
$= \sum_i \mathbf{x}_i^T\mathbf{x}_i - \sum_i \mathbf{u}^T\mathbf{x}_i\mathbf{x}_i^T\mathbf{u}$	
$= \sum_i \mathbf{x}_i^T\mathbf{x}_i - \mathbf{u}^T\left(\sum_i \mathbf{x}_i\mathbf{x}_i^T\right)\mathbf{u} = \sum_i \mathbf{x}_i^T\mathbf{x}_i - \mathbf{u}^T\alpha\mathbf{u}$	Recall $\left(\sum_i \mathbf{x}_i\mathbf{x}_i^T\right)\mathbf{u} = \alpha\mathbf{u}$
$= \sum_i \mathbf{x}_i^T\mathbf{x}_i - \alpha\mathbf{u}^T\mathbf{u}$	Recall $\mathbf{u}^T\mathbf{u} = 1$
$= \sum_i \mathbf{x}_i^T\mathbf{x}_i - \alpha$	Choose the largest α to minimize this error

- find more basis
  - Get the "error" signal $\mathbf{xx} - \mathbf{uu1uu1}TT\mathbf{xx}$ by subtracting the first-level approximation from the original signal.
  - Treat the "error signal" as a new signal, and repeat the estimation on the "error" signal.
  - Get the second-level "error" by subtracting the scaled second typical basis from the first-level error.
  - Repeat the estimation on the second-level "error" signal.
  - We can continue the process until we find NN basis vectors.

- Under-complete vs over-complete

Suppose we represent the signal $\mathbf{x}$ using N basis vectors as

$\mathbf{x} = \sum_{k=1}^{N} w_k\mathbf{u}_k$. Rewrite it into matrix form as $\mathbf{x} = [\mathbf{u}_1, \mathbf{u}_2, \cdots, \mathbf{u}_N]\begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_N \end{bmatrix}$

Input data dimension $\mathrm{L} \times 1$

Basis vectors, each dimension $\mathrm{L} \times 1$

Number of basis vectors

- When #(Basis vectors) = dim(Input data), $N = L$, exact reconstruction
- When #(Basis vectors) < dim(Input data), $N < L$, under-complete, sparse
- When #(Basis vectors) > dim(Input data), $N > L$, over-complete, redundancy

While techniques such as Principal Component Analysis (PCA) allow us to learn a complete set of basis vectors efficiently, we wish to learn an **over-complete** set of basis vectors to represent input vectors $\mathbf{x} \in \mathbb{R}^n$ (i.e. such that $k > n$). The advantage of having an over-complete basis is that our basis vectors are better able to capture structures and patterns inherent in the input data. However, with an over-complete basis, the coefficients a_i are no longer uniquely determined by the input vector $\mathbf{x}$.

Under-complete	Over-complete
• Hidden layer has less dimensions than the input layer • Compresses the input • Hidden nodes will be good features for the training distribution.	• Hidden layer has more dimensions than the input layer • No compression in hidden layer. • A higher dimension code helps model a more complex distribution.

- Learning(Learn signal representation as neural network)

- Stacked signal representation learning

  - 1.Train the first-stage auto encoder.
  - 2.After training,remove the decoder layer,construct a new autoencoder by taking the latent representation of the previous autoencoder as input.
  - 3.Train the new autoencoder.Note the weights and bias of the encoder from the previously trained autoencoders are fixed when training the newly constructed autoencoder.
  - 4.Repeat steps2 and 3 until enough layers are trained.
- Feature embedding: The embedded features are used for other machine learning tasks (e.g., classification).
- Anomaly detection: The embedded features are assumed to be ‘common’ feature of such training dataset, which serves as the reference for anomaly detection.

- Supervised
  - Labels available for both normal data and anomalies
  - Similar to classification with high class imbalance
- Unsupervised
  - No labels assumed
  - Based on the assumption that anomalies are very rare compared to normal data
  - Train an auto-encoder on the training data.
  - Evaluate it on the validation data and the reconstructed error plot.
  - Choose a threshold, which determines whether a value is an outlier (anomalies) or not. This threshold can by dynamic and depends on the previous errors (moving average, time component)
- Label
  - Each test instance is given a normalor anomaly label
  - Typical output of classification-based approaches
- Score
  - Each test instance is assigned an anomaly score
    - allows outputs to be ranked
    - requires an additional threshold parameter

- Intelligent sensing system

- pipeline

- Raw Sensor Data
  - Raw (or unprocessed) sensor data is captured by sensors at the front end of the pipeline
  - Vision: Camera sensors take still or video shots of a scene

    > Scene recognition: Identify multiple objects, faces, and people in an image and using that information to determine the likely activity or context.
    >
    > Object recognition: Identify objects in an image and potentially matching them to a pre-existing database of objects that have been captured before.
    >
    > Gesture recognition: Recognize static poses or moving gestures either specific to the hand/arm or the human body.
    >
    > Anomaly detection: Identify if any anomaly occurred which should trigger additional analysis.
    >
    > Video summarization: Summarize the salient aspects of a long video stream. This includes scene changes, key scenarios and objects/characters that are the focus of the video.

  - Audio/Speech: The microphone-like sensors typically capture the audio and analog-to-digital conversion is applied1

    > Audio classification: A common IoT use case for microphones is to classify the environment that the sound was captured in.
    >
    > Voice activity detection: Determine whether there is voice in the captured audio.
    >
    > Speaker recognition: Speaker recognition, voice recognition, attempts to determine who is speaking.
    >
    > Keyword recognition: Recognize whether a particular word was uttered. Keyword recognition can also be generalized to key-phrase recognition and both are typically used as triggers for additional activity such as starting a session of commands or bringing up an application.
    >
    > Command and control: Refers to using a small set of phrases in speech recognition. For illustration, this could include a set of commands to control a toy car such as “move forward,” “move backward,” “go faster,” “go slower,” “turn right/left,” etc.

  - Motion: Inertial sensors typically measure the acceleration and motion
  - BMI sensors: The Brain Machine Interface sensors, like EEG, report the brain activity measurements as activity graphs
  - Other IoT sensors: temperature, humidity, etc.
- Features

Category	Representative features
Geometrical	Edges, lines, line widths, line relationships (e.g., parallel, perpendicular), circles, shapes, size of enclosed area
Structural	Surface area; relative orientation; orientation in vertical and horizontal ground plane
Statistical	Number of surfaces, area and perimeter, mean, variance, kurtosis, skewness, entropy
Spectral	Color coefficients, spectral peaks and lines
Time domain	Pulse characteristics (rise and fall times, amplitude), pulse width, pulse repetition interval, moments
Frequency domain	Fourier coefficients, Wavelet coefficients

- Recognized Entities
  - Complex task involving spatial and temporal analysis of the extracted features to map the aggregate to pre-known entities.
  - For vision, the recognition task may involve classifying the extracted features to recognize shapes like objects and faces.
  - For audio, statistical analysis and classification of features allows for aggregated features to be recognized as words.
- Semantic Relationship
  - The connection between various recognized entities, for example, the semantic relationship between a key and a lock refers to the operation of opening or closing the lock with the key.
  - For audio, the sentence “I like coffee” connects "I" (the subject) with "coffee“ (object) using the predicate "like.“
  - For vision, semantic context of the video can be analyzed based on identified entities (multiple objects, locations, people, and activities) and scoring these entities with respect to their co-occurrence as well as relation to the type of classification scenario of importance.
- Contextual Relationship
  - Understand the recognized data in context. The context can be obtained in multiple ways. These include using other sensors as well as using the history of recognized data for context recognition.
  - For vision: A face recognition surveillance system can identify a potential invasion or normal operation depending on when the recognition takes place (day vs. night) or depending on the person that is recognized.
- Knowledge
  - Knowledge representation: Knowledge needs to be stored and represented in a manner such that the semantic information and relationships between various concepts is retained and is modifiable.
  - Knowledge operations: These include tools, APIs, and programming languages to insert and extract data from the knowledge database. This involves understanding the incoming semantic data and providing a mechanism to find and manipulate the correct relationships. The extraction process involves responding to different queries targeting relationships between different entities and concepts.

- data fusion
  - Brief idea
    - Representation: The information of the fusion process has an abstract level higher than each input data set.
    - Certainty: If V is the sensor data before fusion and $p(V)$ is the a priori probability of the data before fusion, then the gain in certainty is the growth in $p(V)$ after

fusion.

- Accuracy: The standard deviation on the data after the fusion process is smaller than the standard deviation provided directly by the sources.
- Completeness: Bringing new information to the current knowledge on an environment allows a more complete the view on this environment.

- fields
  - Fusion across sensors (same type): A number of temperature sensors measuring the temperature of an object.
  - Fusion across domains (same type, different definition/range): A number of sensors measure the same attribute over a number of different ranges or domains. This arises, for example, in the definition of a temperature scale.
  - Fusion across attributes (different types): A number of sensors measure different quantities associated with the same experimental situation, such as measuring air temperature, pressure and humidity to determine air refractive index.
  - Fusion across time: Current measurements are fused with historical information.
- Methods
  - Uncoupled sensor data fusion/Late fusion

    - The data fusion occurs at the last possible stage in the respective pipelines. The advantage of this method is that the sensor fusion is simple and can be accomplished using existing technology pipelines for recognition. Domain experts like this approach because cross-domain technical knowledge requirements are minimal.
  - Loosely coupled sensor data fusion/Restricted fusion

    - Restricts the recognition search space of mode modality based on the results from the other. In this approach, one recognition pipe helps to set the context for the other. Once that is accomplished, the second recognition pipe need only perform recognition within the established context boundaries, leading to potential savings in compute and higher performance.
  - Tightly coupled sensor data fusion/Early fusion

        - It allows for combination of multiple sensors that appear as a single complex sensor for most of the applications. Data representation at the lower levels of stack is very sensor-specific, so experts are needed to make sensible decisions for merging data from multiple sources. Each sensor model has a different data representation format at the lowest level, making it very hard to come up with a homogenous representation that preserves the information content.
- registration
    - Spatial alignment. Transformation of the local spatial positions to a common coordinate system. The process involves geo-referencing the location and field-of-view of each sensor.
    - Temporal alignment. Transformation of the local times to a common time axis. In many applications, is performed using a dynamic time warping algorithm.
    - example
        - multi-temporal fusion

        - multi-modal fusion

        - multi-view fusion

- Audio

- Video

- data augment
  - Data augmentation can be viewed as an injection of prior knowledge about the invariant properties of the data against certain transformations. Augmented data can cover unexplored input space, prevent overfitting, and improve the generalization ability of a machine learning model.

- Scaling
  - changes the magnitude of the data in a window by multiplying by a random scalar.
- Magnitude
  - filtering changes the magnitude of each sample by convolving the data window with a smooth curve varying around one.
- Jittering
  - is simulating additive sensor noise. These data augmentation methods may increase robustness against multiplicative and additive noise and improve performance.
- Time-warping
  - perturbs the temporal location. By smoothly distorting the time intervals between samples, the temporal locations of the samples can be changed using time-warping.