Lecture 8: Clustering (Part I)

Statistical Methods for Data Science

Yinan Yu

Department of Computer Science and Engineering

Nov 28, 2024

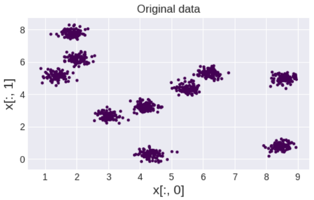
Learning outcome

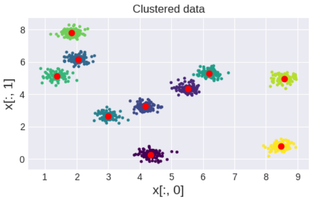
- Understand the difference between supervised learning and unsupervised learning
- Understand how to apply clustering algorithms to the applications discussed in this lecture
- Be able to implement the K-means algorithm
- Be able to explain the within-cluster sum of squared error (SSE) and the Silhouette score
- Be able to determine K and the best initial guesses using SSE and the Silhouette score

Today

Clustering

- Input (left): we start with blobs of data points (original data)

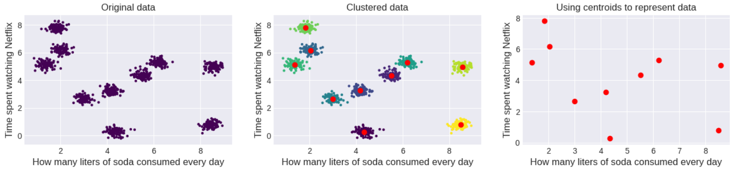




- Output (right): we assign each of these data points to a specific group
- Each group is called a **cluster**
- The process of finding these clusters is called **clustering**
- Clustering is widely used for different purposes; clustering algorithm development **does not require expensive annotations**; clustering is **unsupervised**

Application

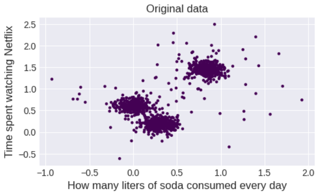
1. To reduce a large amount of data into fewer data points by, e.g., representing a data set with only a few centroids
   Example: you have access to the time people spend on Netflix and the amount of soda they consume everyday; you want to find patterns from this data set

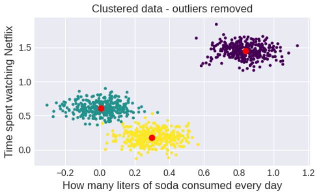


   Group these people into clusters and only use the centroids for data exploration or as input data for downstream analysis
   One important application is the recommender system
   - Task: find patterns of preferred items from a massive number of users
   - Challenge: there are too many users (all data points)
   - Solution: we recommend items to users on a cluster level (only the centroids)

Application (cont.)

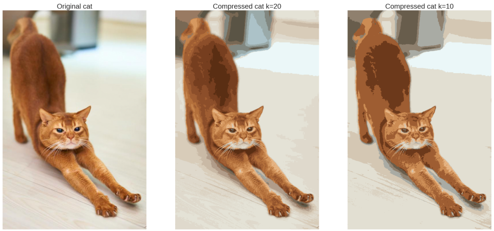
2. To detect and remove **outliers** - data points significantly distant from any of the clusters (here we assume that the clusters represent the underlying model)





Application (cont.)

3. Image compression



- Each data point is a pixel in the image, i.e. $\boldsymbol{x} = [red, green, blue] = [x_1, x_2, x_3]$, where $red, green, blue \in [0, 255]$ integers
- Run clustering algorithms in this RGB color space and find K centroids
- Replace each pixel with its closest centroid
- Now we only use $3 \times K$ unique values to represent the image instead of 3×256 values
- In this example, with $K = 10$ centroids, when we save the .png image, we have a reduction from 328.5 kB to 43.4 kB

Today

Clustering modeling

- Modeling for clustering

$$y = g(x; \theta \mid h)$$

- Clustering:
  - y: **categorical (nominal)**, scalar - each category is called a **cluster**
  - x: typically **continuous numerical**; feature vector $\boldsymbol{x} = [x_1, \cdots, x_d]$ (similar to classification problems in lecture 5)
  - g: **clustering model**, e.g. K-means, Gaussian mixture models, hierarchical clustering models, etc
    There are mainly four categories of clustering models
    - **Centroid clustering** (geometry-based)
    - **Distribution clustering** (probability-based)
    - Density clustering
    - Hierarchical clustering
  - θ (parameters) and h (hyperparameters) depend on g

Parameter estimation

- Clustering models are **unsupervised learning** algorithms
- In unsupervised learning, the parameters are estimated from an **unlabeled data set**, that is, a data set containing only the feature vectors $\{\boldsymbol{x}_1, \cdots, \boldsymbol{x}_N\}$, e.g.

  $\{$, , $\cdots$, $\}$

  where $\boldsymbol{x}_i =$ pixel values in a picture and the task is to group **similar** ducks into the same cluster
- Clustering tasks do not require annotations - it is cheaper, but also more difficult to evaluate because there are no predefined clusters!
- The **similarity** is not uniquely defined
- In this course, we will look at one commonly used parameter estimation technique called the **Expectation-Maximization (EM)** algorithm

Models in this course

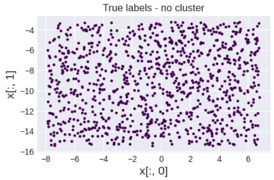
We are going to introduce various categories of clustering techniques; then we focus on two clustering models

- K-means (centroid clustering)
  - **Parameters**: K centroids
  - **Hyperparameters**: the number of centroids K
  - **Parameter estimation**: an iterative method to update the centroids until convergence; this method can be interpreted as a simplified version of the Expectation-Maximization algorithm
- Gaussian mixture models (distribution clustering)
  - **Parameters**: K priors, K Gaussian likelihood (the big two!)
  - **Hyperparameters**: the number of Gaussian components K
  - **Parameter estimation**: the Expectation-Maximization algorithm

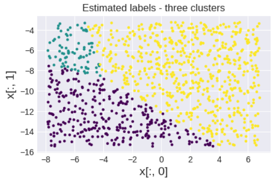
Today

Let's try something out!

- Generate some data $\{[x_1^1, x_2^1], \cdots, [x_1^N, x_2^N]\}$ from a uniform distribution (np.random.uniform) 😃
  - there are **no clusters**



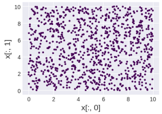
- Run a clustering algorithm

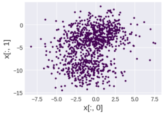


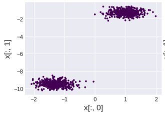
😭

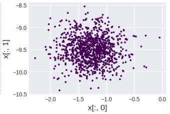
Take one step back: is the data “clusterable”?

- Do you see any clusters in the following plots?









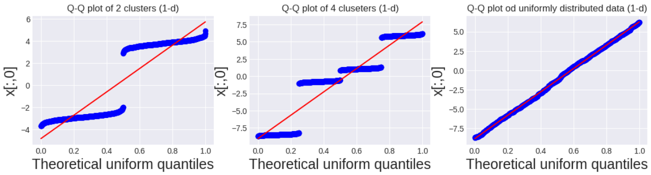
  - Figure 1: data is generated from a uniform distribution - no cluster
  - Figure 2: data is generated from three different Gaussian distributions - three clusters
  - Figure 3: data is generated from two different Gaussian distributions - two clusters
  - Figure 4: data is generated from one Gaussian distribution - one cluster
- How to decide if the data is “culsterable”?
  - Need to define what a cluster is
  - Need to define the “null hypothesis”, i.e. the situation where there are no clusters
- There is no ground truth label - there are various ways of defining these prerequisites, which makes it a difficult task!
- Now spend 30 secs staring at the plots and try to think how you can measure if the data set is clusterable

Clustering tendency

The general idea is to **compare** the **data distribution** with a **theoretical distribution with no clustering tendency**!

Let $\boldsymbol{x}_i = \left[x_1^i, \quad \cdots, \quad x_d^i\right]$ be a feature vector in this course, when we need to index both the dimension of the feature vector and the data point, we use a superscript to index the data point and a subscript to index the dimension

- **For example**, we can make a qq-plot to compare the set $\{x_j^1, \cdots, x_j^N\}$ and a non-clusterable theoretical probability distribution, e.g. a uniform distribution



- We can repeat this for all dimensions $j = 1, \cdots, d$
- But then the question is how to aggregate all these d dimensions? - Not easy!
- **Comparing distributions gets trickier when $d >> 1$!**
- Approximation and sampling

CHALMERS | GÖTEBORGS UNIVERSITET

Distance measure

- Measures how "similar" two data points are
- The most commonly used distance is the Euclidean distance
- Example: let $\boldsymbol{x} = [x_1, x_2, x_3]$ and $\boldsymbol{y} = [y_1, y_2, y_3]$ be two feature vectors, the Euclidean distance is defined as

$$d(\boldsymbol{x}, \boldsymbol{y}) = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2 + (x_3 - y_3)^2}$$

- A one dimensional value (scalar) computed from high dimensional data (here $d = 3$)
- Pairwise distance
  - Distances between all pairs of data points from two sets
  - Example: let $\{\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3\}$ and $\{\boldsymbol{y}_1, \boldsymbol{y}_2\}$ be two sets, the pairwise distance is defined as

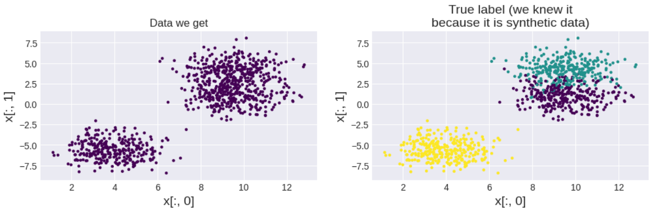
$$\{d(\boldsymbol{x}_1, \boldsymbol{y}_1), d(\boldsymbol{x}_1, \boldsymbol{y}_2), d(\boldsymbol{x}_2, \boldsymbol{y}_1), d(\boldsymbol{x}_2, \boldsymbol{y}_2), d(\boldsymbol{x}_3, \boldsymbol{y}_1), d(\boldsymbol{x}_3, \boldsymbol{y}_2)\}$$

- Generally useful concept
- Pairwise distance can be used to identify clustering tendency by comparing the **distribution of pairwise distances computed from data** to pairwise distances computed from a distribution without clustering tendency, e.g. a **uniform distribution**

Today

K-means

- **Data**: d dimensional feature vector $\boldsymbol{x}$ (in this example $d = 2$)



- **Target** (the coloring in this image - for each $\boldsymbol{x}$, we would like to assign a color to it):

$$y = \arg \min_{k \in \{1, \cdots, K\}} dist(\boldsymbol{x}, \boldsymbol{\mu}_k)$$

  where $dist(\cdot, \cdot)$ is a distance measure; in this course, we use the Euclidean distance (cf. page 16)
- **Parameters**: K centroids $\hat{\boldsymbol{\mu}}_k$
- **Hyperparameters**: K
- **Parameter estimation**: an iterative method to update the centroids until convergence
- It is a **hard clustering** technique - one data point $\boldsymbol{x}$ is assigned to only one cluster y

K-means parameter estimation algorithm to find $\boldsymbol{\mu}_k$

- Initialization: **Randomly choose K centroids** $\boldsymbol{\mu}_k$ for $k = 1, \cdots, K$, e.g. randomly choose K data points from the data set $\mathcal{X}$
- Repeat the two steps below until **convergence**, e.g. $\hat{\boldsymbol{\mu}}_k$ does not change anymore
  - Step 1: For all $i = 1, \cdots, N$, assign $\boldsymbol{x}_i$ to a cluster $\hat{k}_i$ by computing

$$\hat{k}_i = \arg \min_{k \in \{1, \cdots, K\}} dist(\boldsymbol{x}_i, \hat{\boldsymbol{\mu}}_k)$$

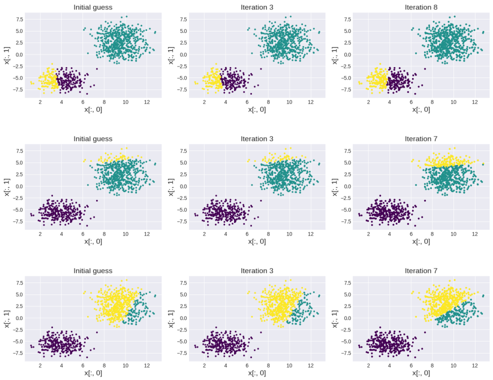
  - Step 2: Let $\mathcal{X}_k$ be the set of all $\boldsymbol{x}_i$ assigned to cluster k and N_k is the size of $\mathcal{X}_k$, compute

$$\hat{\boldsymbol{\mu}}_k \leftarrow \frac{1}{N_k} \sum_{\boldsymbol{x}_j \in \mathcal{X}_k} \boldsymbol{x}_j$$

- There is some **randomness** in the algorithm - we should always be careful when there is randomness

K-means initial guess

Different initializations result in different clusters



A typical solution is to run the algorithm multiple times with different initial points and aggregate the results

K-means parameter estimation pseudocode

```
1: Given a data set $\mathcal{X} = \{\boldsymbol{x}_1, \cdots, \boldsymbol{x}_N\}$
2: Randomly choose $K$ data points from $\mathcal{X}$ as the centroids $\boldsymbol{\mu}_k$ for
   $k = 1, \cdots, K$
3: while true do
4:     Assign $\boldsymbol{x}_i$ to the closest $\boldsymbol{\mu}_k$ for all $i = 1, \cdots, N$
5:     For all $k = 1, \cdots, K$, compute $\boldsymbol{\mu}_k^{new}$ as the center of all $\boldsymbol{x}_i$ assigned
       to cluster $k$
6:     if $\boldsymbol{\mu}_k^{new} == \boldsymbol{\mu}_k$ for all $k$ then
7:         break
8:     else
9:         $\boldsymbol{\mu}_k \leftarrow \boldsymbol{\mu}_k^{new}$
10:    end if
11: end while
```

K-means: pros and cons

- Pros:
  - Convergence guaranteed
  - Easy to implement
  - Scale to large data sets
- Cons - **potential improvement**:
  - Need to choose the hyperparameter K manually - **gradually increase K and monitor the loss during parameter estimation**
  - Dependence on random initial values - **multiple initial values**
  - Do not work well on very high dimensional data - **apply dimensionality reduction techniques before clustering**
  - Not robust to outliers - **try to remove “obvious” outliers before clustering**

Two main challenges for K-means

- **Challenges**:
  - How to choose the hyperparameter K?
  - K-means is sensitive to the initialization of $\hat{\boldsymbol{\mu}}_k$ for $k = 1, \cdots, K$
- **Solution** (for both challenges):
  - Choose a set of **candidate values**, e.g. for the first problem, we can choose $K \in \{1, \cdots, 10\}$; for the second problem, we can randomly select 100 different initial guesses for $\hat{\boldsymbol{\mu}}_k$
  - For each of these **candidate values**, we run the K-means algorithm to estimate the parameters and evaluate the **quality** of the clusters produced by these parameters
  - Choose the **candidate value** that gives the best **quality**
- **Quality** evaluation criteria
  - Within-cluster sum of squared errors (SSE)
  - Silhouette score

Cluster quality evaluation criterion 1: SSE

1. **Within-cluster sum of squared errors (SSE)**: defied as the summation of the distances from all the data points to their closest centroid

$$SSE = \sum_{k=1}^{K} \sum_{\boldsymbol{x} \in C_k} dist(\boldsymbol{x}, \hat{\boldsymbol{\mu}}_k)^2 \tag{1}$$

where C_k denote cluster k; $dist(\cdot,\cdot)$ is a distance measure (**Euclidean distance**: $dist(\boldsymbol{x}, \hat{\boldsymbol{\mu}}_k)^2 = (\boldsymbol{x} - \hat{\boldsymbol{\mu}}_k)^T(\boldsymbol{x} - \hat{\boldsymbol{\mu}}_k)$ for column vectors $\boldsymbol{x}$ and $\hat{\boldsymbol{\mu}}_k$)
Example:

- Given $\boldsymbol{x}_1 = [x_1^1, x_2^1]$, $\boldsymbol{x}_2 = [x_1^2, x_2^2]$, $\boldsymbol{x}_3 = [x_1^3, x_2^3]$, $\boldsymbol{x}_4 = [x_1^4, x_2^4]$, where $\boldsymbol{x}_1, \boldsymbol{x}_2 \in$ cluster 1 with centroid $\boldsymbol{\mu}_1 = [\mu_1^1, \mu_2^1]$; $\boldsymbol{x}_3, \boldsymbol{x}_4 \in$ cluster 2 with centroid $\boldsymbol{\mu}_2 = [\mu_1^2, \mu_2^2]$
- The SSE is computed as

$$\begin{aligned} SSE &= \text{distance in cluster 1} + \text{distance in cluster 2} \\ &= \underbrace{(x_1^1 - \mu_1^1)^2 + (x_2^1 - \mu_2^1)^2}_{dist(\boldsymbol{x_1}, \boldsymbol{\mu_1})^2} + \underbrace{(x_1^2 - \mu_1^1)^2 + (x_2^2 - \mu_2^1)^2}_{dist(\boldsymbol{x_2}, \boldsymbol{\mu_1})^2} \\ &\quad + \underbrace{(x_1^3 - \mu_1^2)^2 + (x_2^3 - \mu_2^2)^2}_{dist(\boldsymbol{x_3}, \boldsymbol{\mu_2})^2} + \underbrace{(x_1^4 - \mu_1^2)^2 + (x_2^4 - \mu_2^2)^2}_{dist(\boldsymbol{x_4}, \boldsymbol{\mu_2})^2} \end{aligned}$$

Cluster quality evaluation criterion 1: SSE (cont.)

1. **Within-cluster sum of squared errors (SSE)** (cont.):

$$SSE = \sum_{k=1}^{K} \sum_{\boldsymbol{x} \in C_k} dist(\boldsymbol{x}, \hat{\boldsymbol{\mu}}_k)^2$$

- SSE is essentially an error term: we want SSE to be small - choose the K value that minimizes SSE?
- No can do! $SSE \to 0$ for $K \to N$, i.e. when every data point is their own centroid, $SSE = 0$, which is not optimal - we can't simply choose the K value that corresponds to the smallest SSE
- Instead, the best K is defined as the **elbow** point of the SSE (instead of the minimum), i.e. **the point with the maximum curvature** - find the largest K where the SSE does not go down significantly by further increasing K
- This method is also called the **elbow method** (in Python, you can find a library to compute the elbow point)

Cluster quality evaluation criterion 2: Silhouette score

2. **Silhouette score** S: the idea is that a good clustering should have **compact clusters** with a **large separation between different clusters**. This is characterized by the **within-cluster distance** and **between-cluster distance**

Cluster quality evaluation criterion 2: Silhouette score (cont.)

2. **Silhouette score S** (cont.):
   Example: given data $x_1, x_2, x_3 \in C_1$, $x_4, x_5 \in C_2$, $x_6, x_7 \in C_3$; $K = 3$; C_k denotes the set of cluster k; $|C_k|$ is the cardinality (size) of the set C_k
   - **Within-cluster distance**: measures how data points scatter in relation to $\boldsymbol{x}_i$ within its own cluster; let $\boldsymbol{x}_i$ be a data point from cluster k,

     $$a_i = \frac{1}{|C_k| - 1} \sum_{\boldsymbol{x}_j \in C_k \text{ and } j \neq i} dist(\boldsymbol{x}_i, \boldsymbol{x}_j)$$

     In this example, let $i = 1$, $x_1 \in C_1$; there are $|C_1| = 3$ data points in cluster 1

     $$a_1 = \frac{1}{3-1}\left(dist(x_1, x_2) + dist(x_1, x_3)\right)$$

   - **Between-cluster distance**: measures how data points scatter in relation to $\boldsymbol{x}_i$ when these data points are from other clusters

     $$b_i = \min_{k' \neq k, k' \in \{1, \cdots, K\}} \frac{1}{|C_{k'}|} \sum_{\boldsymbol{x}_j \in C_{k'}} dist(\boldsymbol{x}_i, \boldsymbol{x}_j)$$

     In the example, $|C_2| = |C_3| = 2$

     $$b_1 = \min\left(\frac{1}{2}\left(dist(x_1, x_4) + dist(x_1, x_5)\right), \frac{1}{2}\left(dist(x_1, x_6) + dist(x_1, x_7)\right)\right)$$

Cluster quality evaluation criterion 2: Silhouette score (cont.)

2. **Silhouette score S** (cont.):
    - Silhouette score for one data point $\boldsymbol{x}_i$:

$$S_i = \begin{cases} \frac{b_i - a_i}{\max(a_i, b_i)}, & \text{if } |C_k| > 1 \\ 0, & \text{if } |C_k| = 1 \end{cases}$$

A large S_i indicates a compact cluster k in relation to $\boldsymbol{x}_i$ and a large distance from $\boldsymbol{x}_i$ to clusters other than k

    - Silhouette score for the data set

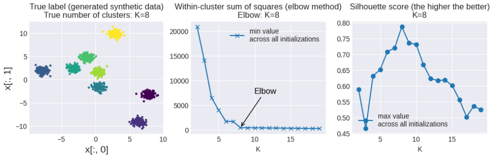
$$S = \frac{1}{N} \sum_{i=1}^{N} S_i, \quad S \in [-1, 1]$$

    - A large Silhouette score indicates a good clustering quality

Example - choose K

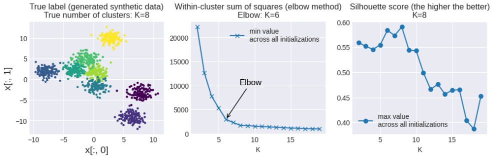
Clusters with equal variance ($K = 8$)

- SSE: $K = 8$
- Silhouette score: $K = 8$



Overlapping clusters with unequal variances ($K = 8$)

- SSE: $K = 6$
- Silhouette score: $K = 8$; but $K = 6$ and $K = 8$ have similar Silhouette scores

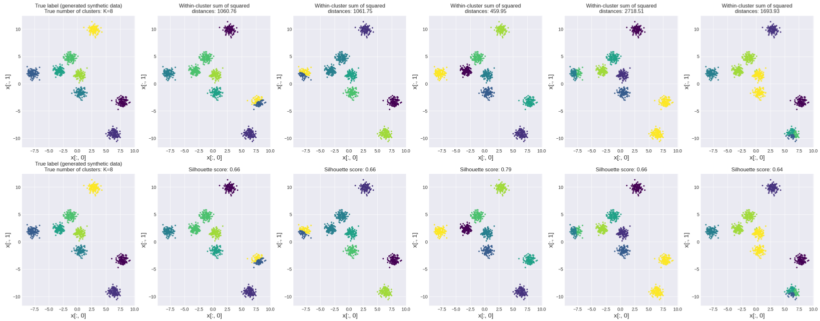


Example - choose initial guess

- Each column corresponds to a different initialization
- For a given K, choose the initialization that gives the smallest SSE or largest Silhouette score

Clusters with equal variance ($K = 8$)

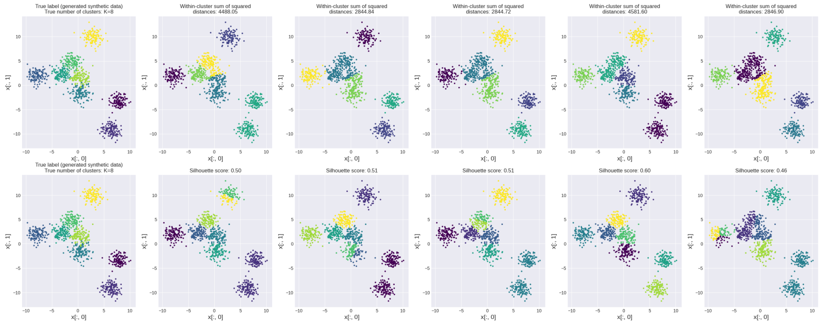
- SSE: $K = 8$
- Silhouette score: $K = 8$



Example - choose initial guess (cont.)

Overlapping clusters with unequal variances ($K = 8$)

- SSE: $K = 6$
- Silhouette score: $K = 8$



Today

Summary

So far:

- Data types and data containers
- Descriptive data analysis: descriptive statistics, visualization
- Probability distributions, events, random variables, PMF, PDF, parameters
- CDF, Q-Q plot, how to compare two distributions (data vs theoretical, data vs data)
- Modeling
- Parameter estimation: maximum likelihood estimation (MLE) and maximum a posteriori estimation (MAP)
- Classification, multinomial naive Bayes classifier, Gaussian naive Bayes classifier
- Clustering, clustering tendency
- Centroid clustering, k-means, parameter estimation, SSE, Silhouette score

Next:

- Gaussian Mixture Models (GMMs)

Before next lecture:

- Gaussian distribution
- The Bayes' rule

You will never get rid of me!