6. Clustering

6.1 Unsupervised Learning

Unsupervised learning is a type of algorithm that learns patterns from untagged data. The hope is that through mimicry, which is an important mode of learning in people, the machine is forced to build a compact internal representation of its world and then generate imaginative content from it. In contrast to supervised learning where data is tagged by an expert, e.g. as a "ball" or "fish", unsupervised methods exhibit self-organisation that captures patterns as probability densities or a combination of neural feature preferences. The other levels in the supervision spectrum are reinforcement learning where the machine is given only a numerical performance score as guidance, and semi-supervised learning where a smaller portion of the data is tagged. Two broad methods in Unsupervised Learning are Neural Networks and Probabilistic Methods.

阅读后结合上课内容掌握何为非监督学习。

6.2 Cluster Analysis

Cluster analysis or clustering is the task of grouping a set of objects in such a way that objects in the same group (called a cluster) are more similar (in some sense) to each other than to those in other groups (clusters). It is a main task of exploratory data analysis, and a common technique for statistical data analysis, used in many fields, including pattern recognition, image analysis, information retrieval, bioinformatics, data compression, computer graphics and machine learning.

Cluster analysis itself is not one specific algorithm, but the general task to be solved. It can be achieved by various algorithms that differ significantly in their understanding of what constitutes a cluster and how to efficiently find them. Popular notions of clusters include groups with

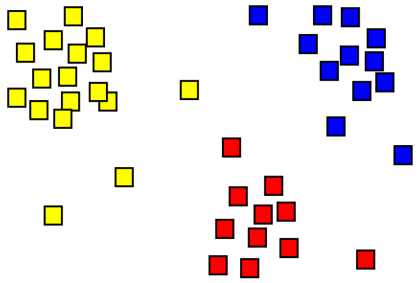

Figure 6.1: The result of a cluster analysis shown as the coloring of the squares into three clusters.

small distances between cluster members, dense areas of the data space, intervals or particular statistical distributions. Clustering can therefore be formulated as a multi-objective optimisation problem. The appropriate clustering algorithm and parameter settings (including parameters such as the distance function to use, a density threshold or the number of expected clusters) depend on the individual data set and intended use of the results. Cluster analysis as such is not an automatic task, but an iterative process of knowledge discovery or interactive multi-objective optimisation that involves trial and failure. It is often necessary to modify data preprocessing and model parameters until the result achieves the desired properties.

Besides the term clustering, there are a number of terms with similar meanings, including automatic classification, numerical taxonomy, botryology, typological analysis, and community detection. The subtle differences are often in the use of the results: while in data mining, the resulting groups are the matter of interest, in automatic classification the resulting discriminative power is of interest.

Cluster analysis was originated in anthropology by Driver and Kroeber in 1932 and introduced to psychology by Joseph Zubin in 1938 and Robert Tryon in 1939 and famously used by Cattell beginning in 1943 for trait theory classification in personality psychology.

阅读后结合上课内容掌握聚类任务的基本定义。

6.3 Hierarchical Clustering

In data mining and statistics, hierarchical clustering (also called hierarchical cluster analysis or HCA) is a method of cluster analysis which seeks to build a hierarchy of clusters. Strategies for hierarchical clustering generally fall into two types:

- Agglomerative: This is a 'bottom-up' approach: each observation starts in its own cluster,

and pairs of clusters are merged as one moves up the hierarchy.

- Divisive: This is a 'top-down' approach: all observations start in one cluster, and splits are performed recursively as one moves down the hierarchy.

In general, the merges and splits are determined in a greedy manner. The results of hierarchical clustering are usually presented in a dendrogram.

The standard algorithm for hierarchical agglomerative clustering (HAC) has a time complexity of $\mathcal{O}(n^3)$ and requires $\Omega(n^2)$ memory, which makes it too slow for even medium data sets. However, for some special cases, optimal efficient agglomerative methods (of complexity $\mathcal{O}(n^2)$ are known: SLINK for single-linkage and CLINK for complete-linkage clustering. With a heap, the runtime of the general case can be reduced to $\mathcal{O}(n^2 \log n)$, an improvement on the aforementioned bound of $\mathcal{O}(n^3)$, at the cost of further increasing the memory requirements. In many cases, the memory overheads of this approach are too large to make it practically usable.

Except for the special case of single-linkage, none of the algorithms (except exhaustive search in $\mathcal{O}(2^n)$ can be guaranteed to find the optimum solution.

Divisive clustering with an exhaustive search is $\mathcal{O}(2^n)$, but it is common to use faster heuristics to choose splits, such as k-means.

阅读后结合上课内容掌层次聚类的基本算法思想。

6.4 Metric

The choice of an appropriate metric will influence the shape of the clusters, as some elements may be relatively closer to one another under one metric than another. For example, in two dimensions, under the Manhattan distance metric, the distance between the origin (0,0) and (0.5, 0.5) is the same as the distance between the origin and (0, 1), while under the Euclidean distance metric the latter is strictly greater.

Some commonly used metrics for hierarchical clustering are:

- Euclidean distance $\|a-b\|_2 = \sqrt{\sum_i (a_i - b_i)^2}$
- Squared Euclidean distance $\|a-b\|_2^2 = \sum_i (a_i - b_i)^2$
- Manhattan (or city block) distance $\|a-b\|_1 = \sum_i |a_i - b_i|$
- Maximum distance (or Chebyshev distance) $\|a-b\|_\infty = \max_i |a_i - b_i|$
- Mahalanobis distance $\sqrt{(a-b)^\top S^{-1}(a-b)}$ where S is the Covariance matrix

阅读后结合上课内容掌握常见的度量!

6.5 Single-linkage Clustering

In statistics, single-linkage clustering is one of several methods of hierarchical clustering. It is based on grouping clusters in bottom-up fashion (agglomerative clustering), at each step combining two clusters that contain the closest pair of elements not yet belonging to the same cluster as each other.

A drawback of this method is that it tends to produce long thin clusters in which nearby elements of the same cluster have small distances, but elements at opposite ends of a cluster may be much farther from each other than two elements of other clusters. This may lead to difficulties in defining classes that could usefully subdivide the data.

6.5.1 Overview

In the beginning of the agglomerative clustering process, each element is in a cluster of its own. The clusters are then sequentially combined into larger clusters, until all elements end up being in the same cluster. At each step, the two clusters separated by the shortest distance are combined. The function used to determine the distance between two clusters, known as the linkage function, is what differentiates the agglomerative clustering methods.

In single-linkage clustering, the distance between two clusters is determined by a single pair of elements: those two elements (one in each cluster) that are closest to each other. The shortest of these pairwise distances that remain at any step causes the two clusters whose elements are involved to be merged. The method is also known as nearest neighbour clustering. The result of the clustering can be visualised as a dendrogram, which shows the sequence in which clusters were merged and the distance at which each merge took place.

Mathematically, the linkage function –the distance $D(X,Y)$ between clusters X and Y –is described by the expression

$$D(X,Y) = \min_{x\in X, y\in Y} d(x,y),$$

where X and Y are any two sets of elements considered as clusters, and $d(x,y)$ denotes the distance between the two elements x and y.

6.5.2 Algorithm

The following algorithm is an agglomerative scheme that erases rows and columns in a proximity matrix as old clusters are merged into new ones. The $N \times N$ proximity matrix D contains all distances $d(i,j)$. The clusterings are assigned sequence numbers $0,1,\ldots,n-1$ and $L(k)$ is the level of the k-th clustering. A cluster with sequence number m is denoted (m) and the proximity between clusters (r) and (s) is denoted $d[(r),(s)]$.

The single linkage algorithm is composed of the following steps:

1. Begin with the disjoint clustering having level $L(0) = 0$ and sequence number $m = 0$.

2. Find the most similar pair of clusters in the current clustering, say pair $(r), (s)$, according to $d[(r), (s)] = \min d[(i), (j)]$ where the minimum is over all pairs of clusters in the current clustering.

3. Increment the sequence number: $m = m + 1$. Merge clusters (r) and (s) into a single cluster to form the next clustering m. Set the level of this clustering to $L(m) = d[(r), (s)]$.

4. Update the proximity matrix, D, by deleting the rows and columns corresponding to clusters (r) and (s) and adding a row and column corresponding to the newly formed cluster. The proximity between the new cluster, denoted (r,s) and old cluster (k) is defined as $d[(r,s), (k)] = \min\{d[(k), (r)], d[(k), (s)]\}$.

5. If all objects are in one cluster, stop. Else, go to step 2.

6.6 Complete-linkage Clustering

Complete-linkage clustering is one of several methods of agglomerative hierarchical clustering. At the beginning of the process, each element is in a cluster of its own. The clusters are then sequentially combined into larger clusters until all elements end up being in the same cluster. The method is also known as farthest neighbour clustering. The result of the clustering can be visualised as a dendrogram, which shows the sequence of cluster fusion and the distance at which each fusion took place.

6.6.1 Overview

At each step, the two clusters separated by the shortest distance are combined. The definition of 'shortest distance' is what differentiates between the different agglomerative clustering methods. In complete-linkage clustering, the link between two clusters contains all element pairs, and the distance between clusters equals the distance between those two elements (one in each cluster) that are farthest away from each other. The shortest of these links that remains at any step causes the fusion of the two clusters whose elements are involved.

Mathematically, the complete linkage function —the distance $D(X,Y)$ between clusters X and Y —is described by the following expression : $D(X,Y) = \max_{x \in X, y \in Y} d(x,y)$ where

- $d(x,y)$ is the distance between elements $x \in X$ and $y \in Y$;
- X and Y are two sets of elements (clusters).

6.6.2 Algorithm

The following algorithm is an agglomerative scheme that erases rows and columns in a proximity matrix as old clusters are merged into new ones. The $N \times N$ proximity matrix D contains all distances d(i,j). The clusterings are assigned sequence numbers 0,1,....., (n 1) and L(k) is the level

of the kth clustering. A cluster with sequence number m is denoted (m) and the proximity between clusters (r) and (s) is denoted d[(r),(s)].

The complete linkage clustering algorithm consists of the following steps:

1. Begin with the disjoint clustering having level $L(0) = 0$ and sequence number $m = 0$.

2. Find the most similar pair of clusters in the current clustering, say pair $(r), (s)$, according to $d[(r), (s)] = \min d[(i), (j)]$ where the minimum is over all pairs of clusters in the current clustering.

3. Increment the sequence number: $m = m + 1$. Merge clusters (r) and (s) into a single cluster to form the next clustering m. Set the level of this clustering to $L(m) = d[(r), (s)]$.

4. Update the proximity matrix, D, by deleting the rows and columns corresponding to clusters (r) and (s) and adding a row and column corresponding to the newly formed cluster. The proximity between the new cluster, denoted (r,s) and old cluster (k) is defined as $d[(r), (s)] = \max\{d[(k), (r)], d[(k), (s)]\}$.

5. If all objects are in one cluster, stop. Else, go to step 2.

6.7 UPGMA

UPGMA (unweighted pair group method with arithmetic mean) is a simple agglomerative (bottom-up) hierarchical clustering method. The method is generally attributed to Sokal and Michener.

The UPGMA method is similar to its weighted variant, the WPGMA method.

Note that the unweighted term indicates that all distances contribute equally to each average that is computed and does not refer to the math by which it is achieved. Thus the simple averaging in WPGMA produces a weighted result and the proportional averaging in UPGMA produces an unweighted result (see the working example).

The UPGMA algorithm constructs a rooted tree (dendrogram) that reflects the structure present in a pairwise similarity matrix (or a dissimilarity matrix). At each step, the nearest two clusters are combined into a higher-level cluster. The distance between any two clusters $\mathscr{A}$ and $\mathscr{B}$, each of size (i.e., cardinality) $|\mathscr{A}|$ and $|\mathscr{B}|$, is taken to be the average of all distances $d(x,y)$ between pairs of objects x in $\mathscr{A}$ and y in $\mathscr{B}$, that is, the mean distance between elements of each cluster:

$$\frac{1}{|\mathscr{A}| \cdot |\mathscr{B}|} \sum_{x \in \mathscr{A}} \sum_{y \in \mathscr{B}} d(x,y)$$

In other words, at each clustering step, the updated distance between the joined clusters $\mathscr{A} \cup \mathscr{B}$ and a new cluster X is given by the proportional averaging of the $d_{\mathscr{A},X}$ and $d_{\mathscr{B},X}$ distances:

$$d_{(\mathscr{A} \cup \mathscr{B}),X} = \frac{|\mathscr{A}| \cdot d_{\mathscr{A},X} + |\mathscr{B}| \cdot d_{\mathscr{B},X}}{|\mathscr{A}| + |\mathscr{B}|}$$

The UPGMA algorithm produces rooted dendrograms and requires a constant-rate assumption - that is, it assumes an ultrametric tree in which the distances from the root to every branch tip are

equal. When the tips are molecular data (i.e., DNA, RNA and protein) sampled at the same time, the ultrametricity assumption becomes equivalent to assuming a molecular clock.

阅读后结合上课内容掌握上述三种层次聚类的具体方法，了解其中的核心区别，并能够对应分析出其最终的聚类特性。

6.8 k-means

k-means clustering is a method of vector quantisation, originally from signal processing, that aims to partition n observations into k clusters in which each observation belongs to the cluster with the nearest mean (cluster centres or cluster centroid), serving as a prototype of the cluster. This results in a partitioning of the data space into Voronoi cells. k-means clustering minimises within-cluster variances (squared Euclidean distances), but not regular Euclidean distances, which would be the more difficult Weber problem: the mean optimises squared errors, whereas only the geometric median minimises Euclidean distances. For instance, better Euclidean solutions can be found using k-medians and k-medoids.

The problem is computationally difficult (NP-hard); however, efficient heuristic algorithms converge quickly to a local optimum. These are usually similar to the expectation-maximisation algorithm for mixtures of Gaussian distributions via an iterative refinement approach employed by both k-means and Gaussian mixture modelling. They both use cluster centres to model the data; however, k-means clustering tends to find clusters of comparable spatial extent, while the Gaussian mixture model allows clusters to have different shapes.

The unsupervised k-means algorithm has a loose relationship to the k-nearest neighbour classifier, a popular supervised machine learning technique for classification that is often confused with k-means due to the name. Applying the 1-nearest neighbour classifier to the cluster centres obtained by k-means classifies new data into the existing clusters. This is known as nearest centroid classifier or Rocchio algorithm.

6.8.1 Description

Given a set of observations $(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n)$, where each observation is a d-dimensional real vector, k-means clustering aims to partition the n observations into $k(\leq n)$ sets $\mathbf{S} = S_1, S_2, \ldots, S_k$ so as to minimise the within-cluster sum of squares (WCSS) (i.e. variance). Formally, the objective is to find:

$$\arg\min_{\mathbf{S}} \sum_{i=1}^{k} \sum_{\mathbf{x} \in S_i} \|\mathbf{x} - \mu_i\|^2 = \arg\min_{\mathbf{S}} \sum_{i=1}^{k} |S_i| \operatorname{Var} S_i$$

where μ_i is the mean of points in Si. This is equivalent to minimising the pairwise squared deviations of points in the same cluster:

$$\arg\min_{\mathbf{S}} \sum_{i=1}^{k} \frac{1}{|S_i|} \sum_{\mathbf{x},\mathbf{y}\in S_i} \|\mathbf{x}-\mathbf{y}\|^2$$

The equivalence can be deduced from identity $|S_i| \sum_{\mathbf{x}\in S_i} \|\mathbf{x}-\mu_i\|^2 = \sum_{\mathbf{x}\neq\mathbf{y}\in S_i} \|\mathbf{x}-\mathbf{y}\|^2$. Note that, due to that the total variance is constant, this is equivalent to maximising the sum of squared deviations between points in different clusters (between-cluster sum of squares, BCSS). This deterministic relationship is also related to the law of total variance in probability theory.

6.8.2 Algorithms

The most common algorithm uses an iterative refinement technique. Due to its ubiquity, it is often called "the k-means algorithm"; it is also referred to as Lloyd's algorithm, particularly in the computer science community. It is sometimes also referred to as "naive k-means", because there exist much faster alternatives.

Given an initial set of k means $m_1^{(1)},\ldots,m_k^{(1)}$ (see below), the algorithm proceeds by alternating between two steps:

- Assignment step: Assign each observation to the cluster with the nearest mean: that with the least squared Euclidean distance.[8] (Mathematically, this means partitioning the observations according to the Voronoi diagram generated by the means.)

$$S_i^{(t)} = \left\{ x_p : \left\| x_p - m_i^{(t)} \right\|^2 \leq \left\| x_p - m_j^{(t)} \right\|^2 \ \forall j, 1 \leq j \leq k \right\},$$

  where each x_p is assigned to exactly one $S^{(t)}$, even if it could be assigned to two or more of them.

- Update step: Recalculate means (centroids) for observations assigned to each cluster.

$$m_i^{(t+1)} = \frac{1}{\left|S_i^{(t)}\right|} \sum_{x_j \in S_i^{(t)}} x_j$$

The algorithm has converged when the assignments no longer change. The algorithm is not guaranteed to find the optimum.

The algorithm is often presented as assigning objects to the nearest cluster by distance. Using a different distance function other than (squared) Euclidean distance may prevent the algorithm from converging. Various modifications of k-means such as spherical k-means and k-medoids have been proposed to allow using other distance measures.

6.8.3 Complexity

Finding the optimal solution to the k-means clustering problem for observations in d dimensions is:

- NP-hard in general Euclidean space (of d dimensions) even for two clusters,
- NP-hard for a general number of clusters k even in the plane,
- if k and d (the dimension) are fixed, the problem can be exactly solved in time $O(n^{dk+1})$, where n is the number of entities to be clustered.

Thus, a variety of heuristic algorithms such as Lloyd's algorithm given above are generally used.

The running time of Lloyd's algorithm (and most variants) is $O(nkdi)$, where:

- n is the number of d-dimensional vectors (to be clustered)
- k the number of clusters
- i the number of iterations needed until convergence.

On data that does have a clustering structure, the number of iterations until convergence is often small, and results only improve slightly after the first dozen iterations. Lloyd's algorithm is therefore often considered to be of "linear" complexity in practice, although it is in the worst case superpolynomial when performed until convergence.

- In the worst-case, Lloyd's algorithm needs $i = 2^{\Omega(\sqrt{n})}$ iterations, so that the worst-case complexity of Lloyd's algorithm is superpolynomial.
- Lloyd's k-means algorithm has polynomial smoothed running time. It is shown that for arbitrary set of n points in $[0,1]^d$, if each point is independently perturbed by a normal distribution with mean 0 and variance σ^2, then the expected running time of k-means algorithm is bounded by $O(n^{34}k^{34}d^8\log^4(n)/\sigma^6)$, which is a polynomial in n, k, d and $1/\sigma$.
- Better bounds are proven for simple cases. For example, it is shown that the running time of k-means algorithm is bounded by $O(dn^4M^2)$ for n points in an integer lattice $\{1,\ldots,M\}^d$.

Lloyd's algorithm is the standard approach for this problem. However, it spends a lot of processing time computing the distances between each of the k cluster centres and the n data points. Since points usually stay in the same clusters after a few iterations, much of this work is unnecessary, making the naive implementation very inefficient. Some implementations use caching and the triangle inequality in order to create bounds and accelerate Lloyd's algorithm.

阅读后结合上课内容掌握 k-means 方法! 并能够手动完成小规模任务的整个计算流程。

6.9 Expectation–Maximisation

In statistics, an expectation–maximization (EM) algorithm is an iterative method to find (local) maximum likelihood or maximum a posteriori (MAP) estimates of parameters in statistical models, where the model depends on unobserved latent variables. The EM iteration alternates

between performing an expectation (E) step, which creates a function for the expectation of the log-likelihood evaluated using the current estimate for the parameters, and a maximization (M) step, which computes parameters maximizing the expected log-likelihood found on the E step. These parameter-estimates are then used to determine the distribution of the latent variables in the next E step.

The EM algorithm is used to find (local) maximum likelihood parameters of a statistical model in cases where the equations cannot be solved directly. Typically these models involve latent variables in addition to unknown parameters and known data observations. That is, either missing values exist among the data, or the model can be formulated more simply by assuming the existence of further unobserved data points. For example, a mixture model can be described more simply by assuming that each observed data point has a corresponding unobserved data point, or latent variable, specifying the mixture component to which each data point belongs.

Finding a maximum likelihood solution typically requires taking the derivatives of the likelihood function with respect to all the unknown values, the parameters and the latent variables, and simultaneously solving the resulting equations. In statistical models with latent variables, this is usually impossible. Instead, the result is typically a set of interlocking equations in which the solution to the parameters requires the values of the latent variables and vice versa, but substituting one set of equations into the other produces an unsolvable equation.

The EM algorithm proceeds from the observation that there is a way to solve these two sets of equations numerically. One can simply pick arbitrary values for one of the two sets of unknowns, use them to estimate the second set, then use these new values to find a better estimate of the first set, and then keep alternating between the two until the resulting values both converge to fixed points. It's not obvious that this will work, but it can be proven in this context. Additionally, it can be proven that the derivative of the likelihood is (arbitrarily close to) zero at that point, which in turn means that the point is either a local maximum or a saddle point. In general, multiple maxima may occur, with no guarantee that the global maximum will be found. Some likelihoods also have singularities in them, i.e., nonsensical maxima. For example, one of the solutions that may be found by EM in a mixture model involves setting one of the components to have zero variance and the mean parameter for the same component to be equal to one of the data points.

6.9.1 Description

Given the statistical model which generates a set $\mathbf{X}$ of observed data, a set of unobserved latent data or missing values $\mathbf{Z}$, and a vector of unknown parameters θ, along with a likelihood function $L(\theta;\mathbf{X},\mathbf{Z}) = p(\mathbf{X},\mathbf{Z} \mid \theta)$, the maximum likelihood estimate (MLE) of the unknown parameters is

determined by maximising the marginal likelihood of the observed data

$$L(\theta;\mathbf{X}) = p(\mathbf{X} \mid \theta) = \int p(\mathbf{X},\mathbf{Z} \mid \theta)\, d\mathbf{Z} = \int p(\mathbf{X} \mid \mathbf{Z},\theta) p(\mathbf{Z} \mid \theta)\, d\mathbf{Z}$$

However, this quantity is often intractable since $\mathbf{Z}$ is unobserved and the distribution of $\mathbf{Z}$ is unknown before attaining θ.

The EM algorithm seeks to find the MLE of the marginal likelihood by iteratively applying these two steps:

1. Expectation step (E step): Define $Q(\theta \mid \theta^{(t)})$ as the expected value of the log likelihood function of θ, with respect to the current conditional distribution of $\mathbf{Z}$ given $\mathbf{X}$ and the current estimates of the parameters $\theta^{(t)}$:

$$Q(\theta \mid \theta^{(t)}) = \mathrm{E}_{\mathbf{Z}|\mathbf{X},\theta^{(t)}} [\log L(\theta;\mathbf{X},\mathbf{Z})]$$

2. Maximization step (M step): Find the parameters that maximize this quantity:

$$\theta^{(t+1)} = \arg\max_{\theta} Q(\theta \mid \theta^{(t)})$$

The typical models to which EM is applied use $\mathbf{Z}$ as a latent variable indicating membership in one of a set of groups:

- The observed data points $\mathbf{X}$ may be discrete (taking values in a finite or countably infinite set) or continuous (taking values in an uncountably infinite set). Associated with each data point may be a vector of observations.
- The missing values (aka latent variables) $\mathbf{Z}$ are discrete, drawn from a fixed number of values, and with one latent variable per observed unit.
- The parameters are continuous, and are of two kinds: Parameters that are associated with all data points, and those associated with a specific value of a latent variable (i.e., associated with all data points whose corresponding latent variable has that value).

However, it is possible to apply EM to other sorts of models.

The motivation is as follows. If the value of the parameters θ is known, usually the value of the latent variables $\mathbf{Z}$ can be found by maximising the log-likelihood over all possible values of $\mathbf{Z}$, either simply by iterating over $\mathbf{Z}$ or through an algorithm such as the Viterbi algorithm for hidden Markov models. Conversely, if we know the value of the latent variables $\mathbf{Z}$, we can find an estimate of the parameters θ fairly easily, typically by simply grouping the observed data points according to the value of the associated latent variable and averaging the values, or some function of the values, of the points in each group. This suggests an iterative algorithm, in the case where both θ and $\mathbf{Z}$ are unknown:

- First, initialise the parameters θ to some random values.

- Compute the probability of each possible value of $\mathbf{Z}$, given θ.
- Then, use the just-computed values of $\mathbf{Z}$ to compute a better estimate for the parameters θ.
- Iterate steps 2 and 3 until convergence.

The algorithm as just described monotonically approaches a local minimum of the cost function.

阅读后结合上课内容掌握 EM 方法，掌握其概率建模的依据，并能够认识到其逼近理想分布的原因。 推荐此部分的外部参考资源于 B 站 https://www.bilibili.com/video/BV1So4y1d7fc?spm_id_from=333.337.search-card.all.click