Learning Manuscripts

JY. J ECNU
jy.jiang@outlook.com

November 15, 2017

Contents

Chapter 1

Clustering

1.1 *k*-means

1.1.1 Introduction

k-means clustering is a simple and elegant approach for partitioning a data set s into k distinct, non-overlapping clusters. To perform k-means clustering, we must first specify the desired number of clusters k, then the k-means algorithm will assign each observation to exactly one of the k clusters.

1.1.2 Description

In mathematics, that is we want to decompose $S = \{\boldsymbol{x}^{(j)}\}_{i=1}^{n}, \boldsymbol{x}^{(j)} \in \mathbb{R}^m$ into k class $C_1, C_2, \cdot, C_k$, s.t.

$$S = \bigsqcup_{i=1}^{k} C_i.$$

The idea behind k-means clustering is that a good clustering is one for which the within-cluster variation is as small as possible.
model [1]

$$\min_{\mu^{(1)},\cdots,\mu^{(k)}} \sum_{i=1}^{k}\sum_{j=1}^{n} \chi_i^j \|\boldsymbol{x}^{(j)} - \mu^{(i)}\|_2^2$$

where

$$\chi_i^j = \begin{cases} 1, \text{ if } \|\boldsymbol{x}^{(j)} - \mu^{(\mathbf{i})}\|_2^2 = \min_s \|\boldsymbol{x}^{(j)} - \mu^{(\mathbf{s})}\| \\ 0, \text{ else.} \end{cases}$$

We define the **cost function**

$$J(\mu) = \sum_{i=1}^{k}\sum_{j=1}^{n} \chi_i^j \|\boldsymbol{x}^{(j)} - \mu^{(i)}\|_2^2, \quad \mu = (\mu^{(1)}, \cdots, \mu^{(k)})$$

To minimizes $J(\mu^{(1)}, \cdots, \mu^{(k)})$, we set its derivatives to zero

$$\nabla_\mu J(\mu) = \mathbf{0}.$$

[1] Ref: https://en.wikipedia.org/wiki/k-means_clustering#description
k-means: some methods for classification and analysis of multivariate observations,j. macqueen

then we get

$$\mu^{(i)} = \frac{\sum_{j=1}^{n} \chi_i^j \boldsymbol{x}^{(j)}}{\sum_{j=1}^{n} \chi_i^j}.$$

1.1.3 Algorithm

algorithm:

Do {

1. Initialize cluster centroids $\mu_1, \cdots, \mu_k$ randomly
2. Loop until convergence {

for each i { $\chi_i^j = \begin{cases} 1, \text{ if } \|\boldsymbol{x}^{(j)} - \mu^{(\mathbf{i})}\|_2^2 = \min_s \|\boldsymbol{x}^{(j)} - \mu^{(\mathbf{s})}\| \\ 0, \text{ else.} \end{cases}$ }

for each j $\{\mu^{(i)} = \frac{\sum_{j=1}^{n} \chi_i^j \boldsymbol{x}^{(j)}}{\sum_{j=1}^{n} \chi_i^j}.\}$

}

1.2 EM

1.2.1 Introduction

In statistics, an **Expectation-Maximization** (EM) algorithm is an iterative method to find 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.

1.2.2 Description

Given the statistical model which generates a set $\boldsymbol{x}$ of observed data, a set of unobserved latent data or missing values $\mathbf{Z}$, and a vector of unknown parameters $\boldsymbol{\theta}$, along with a likelihood function $L(\boldsymbol{\theta}|\boldsymbol{x}, \mathbf{Z}) = p(\boldsymbol{x}, \mathbf{Z}|\boldsymbol{\theta})$, the maximum likelihood estimate (MLE) of the unknown parameters is determined by the marginal likelihood of the observed data.

$$L(\boldsymbol{\theta}|\boldsymbol{x}) = p(\boldsymbol{x}|\boldsymbol{\theta}) = \int p(\boldsymbol{x}, \mathbf{Z}|\boldsymbol{\theta}) d\mathbf{Z} \qquad (1\triangleright 2.1)$$

The objective of the algorithm is to find the parameter $\boldsymbol{\theta}$, which maximizes the likelihood function $L(\boldsymbol{\theta}|\boldsymbol{x})$ of the observed $\boldsymbol{x}$. Unfortunately, its quantity is often intractable since the random variable $\mathbf{Z}$ is hidden, if $\mathbf{Z}$ is a sequence of events, so that the number of values grows exponentially with the sequence length, making the exact calculation of the sum extremely difficult.

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

E: Expectation step
Calculate the expected value of the log likelihood function, with respect to the conditional distribution of $\mathbf{Z}$ given $\boldsymbol{x}$ under the current estimate of the parameters $\boldsymbol{\theta}^{(t)}$:

$$Q(\boldsymbol{\theta}|\boldsymbol{\theta}^{(t)}) = \mathrm{E}_{\mathbf{Z}|\boldsymbol{x},\boldsymbol{\theta}^{(t)}} \left[\log L(\boldsymbol{\theta}|\boldsymbol{x}, \mathbf{Z})\right]$$

M: Maximization step
Find the parameters $\boldsymbol{\theta}$ that maximize this quantity:

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

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

1. The observed data points $\boldsymbol{x}$ may be discrete (finite or countably) or continuous. Associated with each data point may be a vector of observations.

2. The missing values (or latent variables) $\mathbf{Z}$ are discrete, drawn from a fixed number of values, and with one latent variable per observed unit.

3. The parameters $\boldsymbol{\theta}$ 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.

1.2.3 Properties

Speaking of an E-step is a bit of a misnomer. What are calculated in the first step are the fixed, data-dependent parameters of the function Q. Once the parameters of Q are known, it is fully determined and is maximized in the second M-step of an EM algorithm.

Although an EM iteration does increase the observed data likelihood function, no guarantee exists that the sequence converges to a maximum likelihood estimator. For multimodal distributions, this means that an EM algorithm may converge to a local maximum of the observed data likelihood function, depending on starting values. A variety of heuristic or metaheuristic approaches exist to escape a local maximum, such as random-restart hill climbing (starting with several different random initial estimates (t)), or applying simulated annealing methods.

EM is especially useful when the likelihood is an exponential family: the E-step becomes the sum of expectations of sufficient statistics, and the M-step involves maximizing a linear function. In such a case, it is usually possible to derive closed-form expression updates for each step, using the Sundberg formula (published by Rolf Sundberg using unpublished results of Per Martin-Lof and Anders Martin-Lof).

The EM method was modified to compute maximum a posteriori (MAP) estimates for Bayesian inference in the original paper by Dempster, Laird, and Rubin.

Other methods exist to find maximum likelihood estimates, such as gradient descent, conjugate gradient, or variants of the GaussNewton algorithm. Unlike EM, such methods typically require the evaluation of first and/or second derivatives of the likelihood function.

1.2.4 Proof of correctness

Expectation-maximization works to improve $Q(\boldsymbol{\theta}|\boldsymbol{\theta}^{(t)})$ rather than directly improving $\log p(\boldsymbol{x}|\boldsymbol{\theta})$. Here is shown that improvements to the former imply improvements to the latter.

For any $\mathbf{Z}$ with non-zero probability $p(\mathbf{Z}|\boldsymbol{x}, \boldsymbol{\theta})$, we can write

$$\log p(\boldsymbol{x}|\boldsymbol{\theta}) = \log p(\boldsymbol{x}, \mathbf{Z}|\boldsymbol{\theta}) - \log p(\mathbf{Z}|\boldsymbol{x}, \boldsymbol{\theta}), \text{ since } p(\boldsymbol{x}, \mathbf{Z}|\boldsymbol{\theta}) = p(\boldsymbol{x}|\boldsymbol{\theta})p(\mathbf{Z}|\boldsymbol{x}, \boldsymbol{\theta})$$

We take the expectation over possible values of the unknown data $\mathbf{Z}$ under the current parameter estimate $\boldsymbol{\theta}^{(t)}$ by multiplying both sides by $p(\mathbf{Z}|\boldsymbol{x}, \boldsymbol{\theta}^{(t)})$ and summing (or integrating) over $\mathbf{Z}$. The left-hand side is the expectation of a constant, so we get

$$\begin{aligned} \log p(\boldsymbol{x}|\boldsymbol{\theta}) &= \sum_{\mathbf{Z}} p(\mathbf{Z}|\boldsymbol{x}, \boldsymbol{\theta}^{(t)}) \log p(\boldsymbol{x}, \mathbf{Z}|\boldsymbol{\theta}) - \sum_{\mathbf{Z}} p(\mathbf{Z}|\boldsymbol{x}, \boldsymbol{\theta}^{(t)}) \log p(\mathbf{Z}|\boldsymbol{x}, \boldsymbol{\theta}) \\ &= Q(\boldsymbol{\theta}|\boldsymbol{\theta}^{(t)}) + H(\boldsymbol{\theta}|\boldsymbol{\theta}^{(t)}) \end{aligned}$$

This last equation holds for any value of $\boldsymbol{\theta}$ including $\boldsymbol{\theta} = \boldsymbol{\theta}^{(t)}$,

$$\log p(\boldsymbol{x}|\boldsymbol{\theta}^{(t)}) = Q(\boldsymbol{\theta}^{(t)}|\boldsymbol{\theta}^{(t)}) + H(\boldsymbol{\theta}^{(t)}|\boldsymbol{\theta}^{(t)})$$

and subtracting this last equation from the previous equation gives

$$\log p(\boldsymbol{x}|\boldsymbol{\theta}) - \log p(\boldsymbol{x}|\boldsymbol{\theta}^{(t)}) = Q(\boldsymbol{\theta}|\boldsymbol{\theta}^{(t)}) - Q(\boldsymbol{\theta}^{(t)}|\boldsymbol{\theta}^{(t)}) + H(\boldsymbol{\theta}|\boldsymbol{\theta}^{(t)}) - H(\boldsymbol{\theta}^{(t)}|\boldsymbol{\theta}^{(t)})$$

However, Gibbs' inequality tells us that $H(\boldsymbol{\theta}|\boldsymbol{\theta}^{(t)}) \geq H(\boldsymbol{\theta}^{(t)}|\boldsymbol{\theta}^{(t)})$, so we can conclude that

$$\log p(\boldsymbol{x}|\boldsymbol{\theta}) - \log p(\boldsymbol{x}|\boldsymbol{\theta}^{(t)}) \geq Q(\boldsymbol{\theta}|\boldsymbol{\theta}^{(t)}) - Q(\boldsymbol{\theta}^{(t)}|\boldsymbol{\theta}^{(t)})$$

In words, choosing $\boldsymbol{\theta}$ to improve $Q(\boldsymbol{\theta}|\boldsymbol{\theta}^{(t)})$ beyond $Q(\boldsymbol{\theta}^{(t)}|\boldsymbol{\theta}^{(t)})$ cannot cause $\log p(\boldsymbol{x}|\boldsymbol{\theta})$ to decrease below $\log p(\boldsymbol{x}|\boldsymbol{\theta}^{(t)})$, and so the marginal likelihood of the data is non-decreasing.

The EM algorithm can be viewed as two alternating maximization steps, that is, as an example of coordinate ascent. Consider the function:

$$\begin{aligned}
F(q,\theta) :&= E_q[\log L(\theta|x,Z)] + H(q) \\
&= E_q[\log L(\theta|x,Z)] + E_q[-\log q(Z)] \\
&= E_q[\log L(\theta|x,Z) - \log q(Z)] \\
&= E_q[\log p(x,Z|\theta) - \log q(Z)] \\
&= E_q[\log(p(Z|x,\theta)p(x|\theta)) - \log q(Z)] \\
&= E_q[\log p(Z|x,\theta) + \log p(x|\theta) - \log q(Z)] \\
&= E_q\left[\log \frac{p(Z|x,\theta)}{q(Z)} + \log p(x|\theta)\right] \\
&= E_q\left[\log \frac{p(Z|x,\theta)}{q(Z)}\right] + \log p(x|\theta) \\
&= E_q\left[\log \frac{p(Z|x,\theta)}{q(Z)}\right] + \log L(\theta|x) \\
&= \sum_z q(z)\, \log \frac{p(z|x,\theta)}{q(z)} + \log L(\theta|x)
\end{aligned}$$

where q is an arbitrary probability distribution over the unobserved data Z and $H(q)$ is the **entropy** [2] of the distribution q. This function can be written as

$$F(q,\theta) = -D_{\mathrm{KL}}\big(q \parallel p_{Z|X}(\cdot|x,\theta)\big) + \log L(\theta|x) = \sum_z q(z)\, \log \frac{p(z|x,\theta)}{q(z)} + \log L(\theta|x)$$

where $p_{Z|X}(\cdot|x;\theta)$ is the conditional distribution of the unobserved data given the observed data x and D_{KL} is the **Kullback–Leibler divergence** [3].

1.2.5 Algorithm

Then the steps in the EM algorithm may be viewed as:
algorithm:

Loop until convergence {

E-step: $q^{(t+1)} = arg\max_q \ F(q,\theta^{(t)})$

M-step: $\theta^{(t+1)} = arg\max_\theta \ F(q^{(t)},\theta)$

}

[2] Shannon defined the entropy of a discrete random variable $\boldsymbol{x}$ with possible values $\{x_1,\cdots,x_n\}$ and probability mass function $p(X)$ as: $H(X) = E[-\log p(X)]$

[3] For discrete probability distributions P and Q, the KullbackLeibler divergence from Q to P is defined to be $D_{\mathrm{KL}}(P\|Q) = -\sum_i P(i)\, \log \frac{Q(i)}{P(i)}$

1.3 *k*-NN

1.3.1 Introduction

k-nearest neighbors algorithm (k-NN) is a non-parametric method used for classification and regression. Its a type of instance-based learning, or lazy learning, where the function is only approximated locally and all computation is deferred until classification. The k-NN algorithm is among the simplest of all machine learning algorithms.

Both for classification and regression, a useful technique can be to assign weight to the contributions of the neighbors, so that the nearer neighbors contribute more to the average than the more distant ones. For example, a common weighting scheme consists in giving each neighbor a weight of $1/d$, where d is the distance to the neighbor.

The neighbors are taken from a set of objects for which the class (for k-NN classification) or the object property value (for k-NN regression) is known. This can be thought of as the training set for the algorithm, though no explicit training step is required.

1.3.2 Description

k-NN is a non-parametric method used for classification and regression. In both cases, the input consists of the k closest training examples in the feature space. The output depends on whether k-NN is used for classification or regression:

- In k-NN classification, the output is a class membership. An object is classified by a majority vote of its neighbors, with the object being assigned to the class most common among its k nearest neighbors (k is a positive integer, typically small).
- In k-NN regression, the output is the property value for the object. This value is the average of the values of its k nearest neighbors.

The training examples are vectors in a multidimensional feature space, each with a class label. The training phase of the algorithm consists only of storing the feature vectors and class labels of the training samples.

In the classification phase, k is a user-defined constant, and an unlabeled vector (a query or test point) is classified by assigning the label which is most frequent among the k training samples nearest to that query point.

A commonly used distance metric for continuous variables is Euclidean distance. For discrete variables, such as for text classification, another metric can be used, such as the overlap metric (or Hamming distance). In the context of gene expression microarray data, for example, k-NN has also been employed with correlation coefficients such as Pearson and Spearman. Often, the classification accuracy of k-NN can be improved significantly if the distance metric is learned with specialized algorithms such as Large Margin Nearest Neighbor or Neighbourhood components analysis.

In mathematics, k-NN can be stated as follow

Suppose that the training samples $(\boldsymbol{x}^{(1)}, \boldsymbol{y}^{(1)}), \ldots, (\boldsymbol{x}^{(m)}, \boldsymbol{y}^{(m)}) \in \mathbb{R}^d \times \mathbf{S}$ is known, $\mathbf{S} = \{s_1, \cdots, s_n\}$ is the class label. $(\boldsymbol{x}, \boldsymbol{y})$ is the unlabeled data need to be classified, that is X can be seen as a random variables in $\mathbf{R}^d$ and Y is unknown. So that

$$X|Y = r \sim P_r,\ r \in \mathbf{S}, P_r \text{ is probability distributions}$$

Given some norm $\|\cdot\|$ on $\mathbb{R}^d$ and a point $x \in \mathbb{R}^d$. Reorder training data

$$(\tilde{\boldsymbol{x}}^{(1)}, \tilde{\boldsymbol{y}}^{(1)}), \ldots, (\tilde{\boldsymbol{x}}^{(m)}, \tilde{\boldsymbol{y}}^{(m)}),\ s.t.\ \|\tilde{\boldsymbol{x}}^{(1)} - \boldsymbol{x}\| \leq \cdots \leq \|\tilde{\boldsymbol{x}}^{(m)} - \boldsymbol{x}\|$$

Define a positive small constant k. For all $\|\tilde{\boldsymbol{x}}^{(i)} - \boldsymbol{x}\| < k$, count the number of times each class appears, that is

$$c_i = Card\{\tilde{\boldsymbol{y}}^{(j)} == s_i : j \in \{j : \|\tilde{\boldsymbol{x}}^{(j)} - \boldsymbol{x}\| < k\}\}, i = 1, \cdots, n$$

get pair $\{(s_1, c_1), \cdots, (s_n, c_n)\}$. The k-NN says that

$$\boldsymbol{y} = s_i,\ s.t.\ n_i = \max_i\{c_i\}$$

Parameter selection

The best choice of k depends upon the data, Generally, larger values of k reduce the effect of noise on the classification, but make boundaries between classes less distinct. A good k can be selected by various heuristic techniques. The special case where the class is predicted to be the class of the closest training sample, i.e. when $k = 1$, is called the nearest neighbor algorithm.

The accuracy of the k-NN algorithm can be severely degraded by the presence of noisy or irrelevant features, or if the feature scales are not consistent with their importance. Much research effort has been put into

selecting or scaling features to improve classification. A particularly popular approach is the use of evolutionary algorithms to optimize feature scaling. Another popular approach is to scale features by the mutual information of the training data with the training classes.

In binary (two class) classification problems, it is helpful to choose k to be an odd number as this avoids tied votes. One popular way of choosing the empirically optimal in this setting is via bootstrap method.

1.3.3 Properties

k-NN is a special case of a variable-bandwidth, kernel density "balloon" estimator with a uniform kernel. A peculiarity of the k-NN algorithm is that it is sensitive to the local structure of the data.

The naive version of the algorithm is easy to implement by computing the distances from the test example to all stored examples, but it is computationally intensive for large training sets. Using an approximate nearest neighbor search algorithm makes k-NN computationally tractable even for large data sets. Many nearest neighbor search algorithms have been proposed over the years; these generally seek to reduce the number of distance evaluations actually performed.

k-NN has some strong consistency results. As the amount of data approaches infinity, the two-class k-NN algorithm is guaranteed to yield an error rate no worse than twice the Bayes error rate. Various improvements to the k-NN speed are possible by using proximity graphs.

For multi-class k-NN classification, Cover and Hart (1967) prove an upper bound error rate of

$$R^* \ \leq \ R_{k-\mathrm{NN}} \ \leq \ R^* \left(2 - \frac{nR^*}{n-1}\right)$$

where R^* is the Bayes error rate (which is the minimal error rate possible), $R_{k-\mathrm{NN}}$ is the k-NN error rate, and n is the number of classes in the problem. For $n = 2$ and as the Bayesian error rate R^* approaches zero, this limit reduces to "not more than twice the Bayesian error rate".

1.3.4 Algorithm

Do {

input training data $\{\boldsymbol{x}^{(i)}, \boldsymbol{y}^{(i)}\}_{i=1}^{m}$, test data $\boldsymbol{x}$,

$(\tilde{\boldsymbol{x}}^{(1)}, \tilde{\boldsymbol{y}}^{(1)}), \ldots, (\tilde{\boldsymbol{x}}^{(m)}, \tilde{\boldsymbol{y}}^{(m)}),\ s.t.\ \|\tilde{\boldsymbol{x}}^{(1)} - \boldsymbol{x}\| \leq \cdots \leq \|\tilde{\boldsymbol{x}}^{(m)} - \boldsymbol{x}\|$

$c_i = Card\{\tilde{\boldsymbol{y}}^{(j)} == s_i : j \in \{j : \|\tilde{\boldsymbol{x}}^{(j)} - \boldsymbol{x}\| < k\}\}, i = 1, \cdots, n$

$\boldsymbol{y} = s_i,\ s.t.\ n_i = \max_i\{c_i\}$

}

1.4 Hierarchical Clustering

1.4.1 Introduction

Hierarchical clustering (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.

In the general case, the complexity of agglomerative clustering is $\mathcal{O}(n^2 \log(n))$, which makes it too slow for large data sets. Divisive clustering with an exhaustive search is $\mathcal{O}(2^n)$, which is even worse. 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.

1.4.2 Description

Cluster dissimilarity

In order to decide which clusters should be combined (for agglomerative), or where a cluster should be split (for divisive), a measure of dissimilarity between sets of observations is required. In most methods of hierarchical clustering, this is achieved by use of an appropriate metric (a measure of distance between pairs of observations), and a linkage criterion which specifies the dissimilarity of sets as a function of the pairwise distances of observations in the sets.

Metric

The choice of an appropriate metric will influence the shape of the clusters, as some elements may be close to one another according to one distance and farther away according to another.

Some commonly used metrics for hierarchical clustering are:

- **Minkowski distance:** $\mathbf{d}(\boldsymbol{x}^{(i)}, \boldsymbol{x}^{(j)}) = \|\boldsymbol{x}^{(i)} - \boldsymbol{x}^{(j)}\|_\alpha$

  Manhattan distance: $\alpha = 1, \quad \mathbf{d}(\boldsymbol{x}^{(i)}, \boldsymbol{x}^{(j)}) = \sum_{k=1}^{n} |x_k^{(i)} - x_k^{(j)}|$

  Euclidean distance: $\alpha = 2, \quad \mathbf{d}(\boldsymbol{x}^{(i)}, \boldsymbol{x}^{(j)}) = \sqrt{\sum_{k=1}^{n} (x_k^{(i)} - x_k^{(j)})^2}$

  Chebyshev distance: $\alpha = \infty, \quad \mathbf{d}(\boldsymbol{x}^{(i)}, \boldsymbol{x}^{(j)}) = \max_{1 \le k \le n} |x_k^{(i)} - x_k^{(j)}|$

  When all the variables have different units or the measurement ranges vary greatly, cannot be directly used Minkowski distance, Each variable should be normalized at the beginning.
- **Mahalanobis distance:** $\mathbf{d}(\boldsymbol{x}^{(i)}, \boldsymbol{x}^{(j)}) = \sqrt{(\boldsymbol{x}^{(i)} - \boldsymbol{x}^{(j)})^T \boldsymbol{\Sigma}^{-1} (\boldsymbol{x}^{(i)} - \boldsymbol{x}^{(j)})}$

Similarity Coefficient

Clustering analysis not only can be used to classify the samples, and also can be used to classify the variables. When it is used in classification, we offen use similarity coefficient to measure the similarity between variables.

we denote c_{ij} as the similarity coefficient of variables x_i and x_j, The definition of c_{ij} generally satisfies the following conditions

i $c_{ij} = \pm 1$ iif $x_i = ax_j + b, a \neq 0, b \in \mathbb{R}$
ii $|c_{ij}| \le 1$
iii $|c_{ij}| = c_{ji}$

The two most commonly used similarity coefficients are

1. Angle Cosine

$$c_{ij} = \cos\theta_{ij} = \frac{(\boldsymbol{x}^{(i)})^T \boldsymbol{x}^{(j)}}{[((\boldsymbol{x}^{(i)})^T \boldsymbol{x}^{(i)})(\boldsymbol{x}^{(j)T} \boldsymbol{x}^{(j)})]^{1/2}}$$

   where $\boldsymbol{x} \in \mathbb{R}^n$ is the observation of variable x, θ_{ij} is the included angle of observation $\boldsymbol{x}^{(i)}$ and $\boldsymbol{x}^{(j)}$.

2. correlation coefficient

$$c_{ij} = \rho_{ij} = \frac{(\boldsymbol{x}^{(i)} - \boldsymbol{\mu}^{(i)})^T (\boldsymbol{x}^{(j)} - \boldsymbol{\mu}^{(j)})}{[(\boldsymbol{x}^{(i)} - \boldsymbol{\mu}^{(i)})^T (\boldsymbol{x}^{(i)} - \boldsymbol{\mu}^{(i)})(\boldsymbol{x}^{(j)} - \boldsymbol{\mu}^{(j)})^T (\boldsymbol{x}^{(j)} - \boldsymbol{\mu}^{(j)})]^{1/2}}$$

Note that, if $\boldsymbol{x}^{(i)}, \boldsymbol{x}^{(j)}$ is the normalized, then the angle cosine is the correlation coefficient.

Hierarchical clustering

Hierarchical clustering is one of the most widely used clustering methods in clustering analysis. Its basic idea is

Assume that $\boldsymbol{x}^{(k)} \in \mathbb{R}^n$ be samples, and denote that $d_{ij} = d(\boldsymbol{x}^{(i)}, \boldsymbol{x}^{(j)})$ be the distance between samples $\boldsymbol{x}^{(i)}$ and $\boldsymbol{x}^{(j)}$, $C_1, C_2, \cdots$ be the classes. D_{rs} be the distance between class C_r and C_s.

1. Make each sample $\boldsymbol{x}^{(k)}$ be a class.
2. Define the distance d_{ij} between samples $\boldsymbol{x}^{(i)}, \boldsymbol{x}^{(j)}$, and the distance D_{rs} between classes C_r, C_s.
3. The nearest two classes are merged into a new class, and calculate the distance between the new class and other classes.
4. Combinate the two nearest classes repeatedly, each time a class is reduced until all the samples are merged into one class.

With different class distance setting, we can get different hierarchical clustering algorithm. The common distance between classes as follow

Distance Between Classes

1. **Minimum**
$$D_{rs} = \min_{\boldsymbol{x}^{(i)} \in C_r, \boldsymbol{x}^{(j)} \in C_s} d_{ij}$$
2. **Maximum**
$$D_{rs} = \max_{\boldsymbol{x}^{(i)} \in C_r, \boldsymbol{x}^{(j)} \in C_s} d_{ij}$$
3. **Mean**
$$D_{rs} = \frac{1}{|C_r||C_s|} \sum_{\boldsymbol{x}^{(j)} \in C_r} \sum_{\boldsymbol{x}^{(j)} \in C_s} d_{ij}$$
4. **Centroid**
$$D_{rs} = \|c_r - c_s\|_2^2, \text{ where } c_r = \frac{1}{|C_r|} \sum_{\boldsymbol{x}^{(i)} \in C_r} \boldsymbol{x}^{(i)}, c_s = \frac{1}{|C_s|} \sum_{\boldsymbol{x}^{(i)} \in C_s} \boldsymbol{x}^{(i)}$$

where d is the chosen metric.

1.4.3 Algorithm

algorithm:

Do {

$C_k = \{\boldsymbol{x}^{(k)}\}, k = 1, \cdots, n$

$\mathbf{D}_{(0)} = [D_{rs}]$

For i From 1 to n-1 {

$D_{rs} = \min_{i,j} \mathbf{D}_{i,j} > 0$

$C_{new} = C_r \sqcup C_s$

$\mathbf{D}_{(i)} = \mathbf{D}_{(i-1)}$ delete r-th,s-th row, column

$\mathbf{D}_{(i)} = \mathbf{D}_{(i)}$ add a row, column at (n-i)-th with $D_{new,i}, D_{i,new}$

relabel $C_k, k = 1, \cdots, n - i$

}

}

Chapter 2

Classification

2.1 Logistic Regression

2.1.1 Introduction

Logistic regression was developed by statistician David Cox in 1958. In statistics, logistic regression is a regression model where the dependent variable is categorical. Logistic regression can be binomial, ordinal or multinomial.

- Binomial or binary logistic regression deals with situations in which the observed outcome for a dependent variable can have only two possible types, "0" and "1".
- Multinomial logistic regression deals with situations where the outcome can have three or more possible types that are not ordered.
- Ordinal logistic regression deals with dependent variables that are ordered.

This work covers the case of a binary dependent variable, that is, where the output can take only two values, "0" and "1". The binary logistic model is used to estimate the probability of a binary response based on one or more predictor variables. It allows one to say that the presence of a risk factor increases the odds of a given outcome by a specific factor.

2.1.2 Description

logistic function

$$h(t) = \frac{1}{1 + e^{-t}}, t \in \mathbb{R}$$

its derivative is

$$h'(t) = (1 - h(t))h(t)$$

its inverse

$$t = \log \frac{h(t)}{1 - h(t)}, h(t) \in (0, 1)$$

Assume that $X \in \mathbb{R}^{m+1}, Y \in \{0, 1\}, \boldsymbol{\theta} = (\theta_0, \theta_1, \cdot, \theta_m)^T$, X, Y be random variable, $\boldsymbol{x} = (x_0, x_1, \cdots, x_m)^T, x_0 = 1, y \in \{0, 1\}$ are samples of X, Y respectively, and all samples are independent,

$$Y|X, \boldsymbol{\theta} \sim P(Y|X, \boldsymbol{\theta})$$

Consider a generalized linear model function parameterized by $\boldsymbol{\theta}$

$$h(\boldsymbol{\theta}^T \boldsymbol{x}) = \frac{1}{1 + e^{-\boldsymbol{\theta}^T \boldsymbol{x}}}$$

If we assume that

$$P(y|\boldsymbol{x}, \boldsymbol{\theta}) = [h(\boldsymbol{\theta}^T \boldsymbol{x})]^y [1 - h(\boldsymbol{\theta}^T \boldsymbol{x})]^{(1-y)}$$

We define the loss functions as

$$\begin{aligned} L(\boldsymbol{\theta}) &= \prod_{i=1}^{n} P(y^{(i)}|\boldsymbol{x}^{(i)}, \boldsymbol{\theta}) \\ &= \prod_{i=1}^{n} [h(\boldsymbol{\theta}^T \boldsymbol{x}^{(i)})]^{y^{(i)}} [1 - h(\boldsymbol{\theta}^T \boldsymbol{x}^{(i)})]^{(1-y^{(i)})} \end{aligned}$$

Tt will be easier to maximize the log likelihood:

$$J(\boldsymbol{\theta}) = \log L(\boldsymbol{\theta}) = \sum_{i=1}^{n} (y^{(i)} \log h(\boldsymbol{\theta}^T \boldsymbol{x}^{(i)}) + (1 - y^{(i)}) \log(1 - h(\boldsymbol{\theta}^T \boldsymbol{x}^{(i)}))$$

which can be maximized by gradient descent.

$$\begin{aligned} \frac{\partial J(\boldsymbol{\theta})}{\partial \theta_j} &= \sum_{i=1}^{n} [y^{(i)} \frac{1}{h(\boldsymbol{\theta}^T \boldsymbol{x}^{(i)})} \frac{\partial h(\boldsymbol{\theta}^T \boldsymbol{x}^{(i)})}{\partial \theta_j} + (1 - y^{(i)}) \frac{1}{1 - h(\boldsymbol{\theta}^T \boldsymbol{x}^{(i)})} \frac{\partial (1 - h(\boldsymbol{\theta}^T \boldsymbol{x}^{(i)}))}{\partial \theta_j}] \\ &= \sum_{i=1}^{n} [y^{(i)} \frac{1}{h(\boldsymbol{\theta}^T \boldsymbol{x}^{(i)})} - (1 - y^{(i)}) \frac{1}{1 - h(\boldsymbol{\theta}^T \boldsymbol{x}^{(i)})}] \frac{\partial h(\boldsymbol{\theta}^T \boldsymbol{x}^{(i)})}{\partial \theta_j} \end{aligned}$$

$$= \sum_{i=1}^{n} \frac{y^{(i)} - h(\boldsymbol{\theta}^T \boldsymbol{x}^{(i)})}{h(\boldsymbol{\theta}^T \boldsymbol{x}^{(i)})(1 - h(\boldsymbol{\theta}^T \boldsymbol{x}^{(i)}))} h(\boldsymbol{\theta}^T \boldsymbol{x}^{(i)})(1 - h(\boldsymbol{\theta}^T \boldsymbol{x}^{(i)})) \frac{\partial \boldsymbol{\theta}^T \boldsymbol{x}^{(i)}}{\partial \theta_j}$$
$$= \sum_{i=1}^{n} (y^{(i)} - h(\boldsymbol{\theta}^T \boldsymbol{x}^{(i)})) x_j^{(i)}$$

that is

$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \sum_{i=1}^{n} (y^{(i)} - h(\boldsymbol{\theta}^T \boldsymbol{x}^{(i)}))(\boldsymbol{x}^{(i)})^T$$

We update $\boldsymbol{\theta}$ by GD

$$\boldsymbol{\theta} = \boldsymbol{\theta} - \alpha \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$$

2.1.3 Algorithm

Loop until convergence {
$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \sum_{i=1}^{n} (y^{(i)} - h(\boldsymbol{\theta}^T \boldsymbol{x}^{(i)}))(\boldsymbol{x}^{(i)})^T$
$\boldsymbol{\theta} = \boldsymbol{\theta} - \alpha \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$
}

2.2 Softmax Regression

In statistics, multinomial logistic regression is a classification method that generalizes logistic regression to multiclass problems. That is, it is a model that is used to predict the probabilities of the different possible outcomes of a categorically distributed dependent variable, given a set of independent variables, which may be real-valued, binary-valued, categorical-valued, etc.

2.2.1 Introduction

Multinomial logistic regression is used when the dependent variable in question is nominal and for which there are more than two categories. Nominal means category, which items that cannot be meaningfully ordered.

Multinomial logistic regression is a particular solution to the classification problem that assumes that a linear combination of the observed features and some problem-specific parameters can be used to determine the probability of each particular outcome of the dependent variable. The best values of the parameters for a given problem are usually determined from some training data.

2.2.2 Description

Assumptions