Development of *mknn* package for simulating missing data completion

Group member: Wandi Xiong, Zerui Zhang, Yulu Chen

Introduction

Missing data completion is a challenging research topic in data mining, machine learning and challenging research topic. Missing data completion is widely used to information obtain, inference and prediction in uncertain information condition. K-neighbor algorithm is a common algorithm for missing data. Compared with previous algorithm (Expectation Maximization algorithm, Bayesian method, tolerance relation and so on), K-neighbor algorithm is one of the most effective and a kind of relatively low complexity methods. K-neighbor algorithm is a common algorithm for missing data. Compared with previous algorithm (Expectation Maximization algorithm, Bayesian method, tolerance relation and so on), K-neighbor algorithm is one of the most effective and a kind of relatively low complexity methods. In our package, our team adopted data bias and threshold value to improve K-neighbor algorithm.

Methods

-K-neighbor algorithm completion

Two data, which have the nearest distance, are the most similar. So if a set of sample data misses the dominated variables, you can use the distance between supporting variables in the date set to find the k nearest neighbor samples. The missing data can be replaced by the neighbor supporting data. Using Euclidean distance judge distance between sample points, as shown below.

$$d(X_i, X_j) = \sqrt{\sum_{r=1}^{m} (x_{ir} - x_{jr})^2}$$

In this equation, $X_i = \{x_{i1}, x_{i2}, x_{i3}, \dots, x_{ik}\}$ represents the preceding m dimensionality data and x_{ir} represents the rth dimensionality data.

-Improve K-neighbor algorithm completion

When K =4, namely choose four neighboring data around the missing data, all the neighboring points are in the first quadrant. It shows that there exists a data bias. If only using the K-neighbor algorithm, it may cause larger deviation.

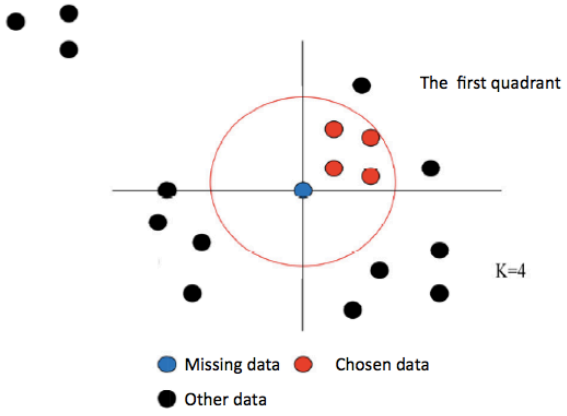

Fig.1 The choice of neighbor points

For missing data $Z_i = \{x_{i1}, x_{i2}, x_{i3}, \ldots, x_{im}, y\}$, preceding m dimensionality data are intact while the $(m+1)^{th}$ dimensionality data is missing. Establish m dimensional space coordinate system based on the m dimensionality data. The space is divided into 2^m quadrants. The nearest data could be found in each dimension and the total number of nearest data in different dimension is set to n, $n \leq 2^m$. When m = 2, n = 4, the result as shown below.

Fig.2 The result of choice

However, sometimes this improved method may lead new problem. When m = 2, n = 4, the distance of the nearest data in second quadrant is longer than other quadrants. It is unreasonable if nearest data chosen from the second quadrant. So in this case, a threshold L is introduced. If d (distance) > L, the data is rejected.

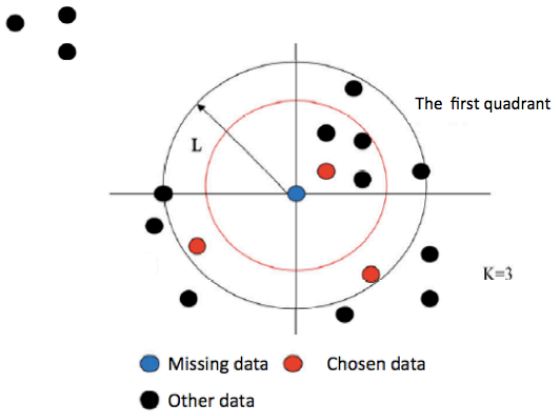

Fig.3 Choice after the perfect.

Because the distances of neighbor data are different, the effects of each neighbor data are also different. Weight function was established to represent the effects of neighbor data on missing data.

$$d(x_i,q)=\sqrt{\sum_{r=1}^{m}(x_{ir}-q_r)^2}$$

$$t(x_i,q)=\frac{1}{d(x_i,q)}$$

$$w(x_i,q)=\frac{t(x_i,q)}{\sum_{i=1}^{k}t(x_i,q)}$$

$$value=\sum_{i=1}^{k}w(x_i,q)\cdot Z_{i(m+1)}$$

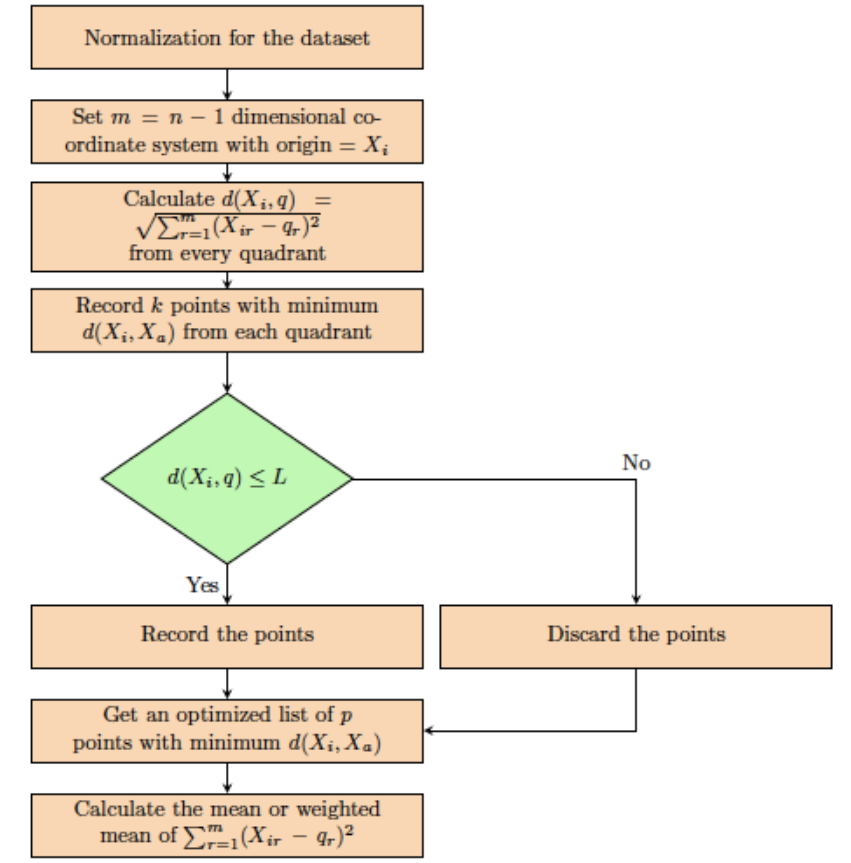

Fig.4 Algorithm flow chart

Results

We used online dataset (https://github.com/paulhendricks/titanic/tree/master/data) to test the improved K-neighbor algorithm. This dataset predicts survival on the Titanic and get familiar with ML basics. In the test, the dominant variable is "Survived". Other variables ("Pclass", "age", "fare", "numsib" and "numpar") are supporting data. We randomly chose some "Survived" data to be the missing data.

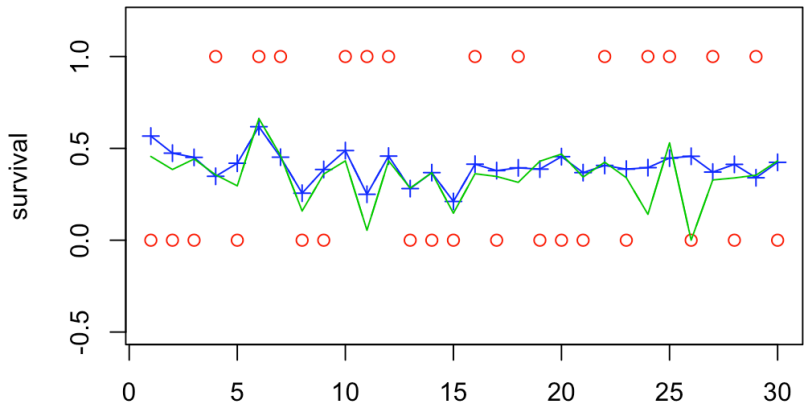

Fig.5 Tracking effect comparison of two methods

The x-axis is the size of sample and the y-axis is the survival value. Red point is the true value of survival. Blue line is the K- neighbor method result while green line is the improved K-neighbor method result.

$$MXRE = max_{i=1}^{k} |\hat{y}_i - y_i|$$

$$MRE = \frac{1}{k}\sum_{i=1}^{k} |\hat{y}_i - y_i|$$

Method	MXRE	MRE
KNN	0.946	13.745
Modified KNN	0.75	14.3

Tab.1 MXRE and MRE value of different KNN methods

Compared with original KNN method, modified KNN method is better than original KNN methods

Readme for mknn

-Description

Using improve K-neighbor algorithm to calculate the missing data. There are

several methods to calculate, depends on whether consider weight and the methods to calculate distance.

-Usage

1. knn_imputation (dataset, k=0, method='weighted', distance='euclidean', L=0)
2. get_quad_val(dataset, posNA, L, k, method)
3. get_estimate_value (method, dataset, d_order, posNA, dm)
4. intercept_L (quad_order, L)
5. intercept_k <- function(ds, k)
6. get_dist_matrix (ds, posNA)
7. get_quad_order (ds, posNA, k)
8. get_quad_list(sqrtSum, quad_vec)
9. isEqual(row1, row2)

sample arguments

source("knn_imputation.R)

res <- knn_imputation(data, k=3, distance="quad")

res<- knn_imputation(data, k=3, method = “weighted”, distance="quad")

-Arguments

k: the number of closest elements used (If k < 1 or k > number of rows in dataset, k will be set using the number of rows in dataset)

method: When method == 'weighted', the function used distance weight to compute the weighted average. Otherwise, use unweighted average

distance: When distance == "euclidean", use classic Euclidean method to compute distance. When distance == "quad", divide quadrants according to each incomplete data point

L: the longest distance that takes into consideration

If L == 0, consider all the complete points

If L > 0, use given L. Return NULL if L is too small

If L < 0, L = median of all complete points

-Value

knn_imputation: Compute the knn imputation values for dataset.

get_quad_val: Get the estimate value when using quad distance.

get_estimate_value: Get the estimate value when using euclidean distance.

intercept_L: Only maintain the data points that are nearer than L.

intercept_k: Extract the closest k data points for each incomplete data point.

get_dist_matrix: Use standard euclidean method to compute distance.

get_quad_order: Divide quadrants and get close data points for each incomplete data point.

get_quad_list: Get the lists of quadrants and divide data.

isEqual: Compare row1 and row2, Return TRUE if row1 == row2.