

Data Mining

CFMV

Application of clustering for finding missed values

Warszawa, November, 26th 2015

Students:

David Miguel Lozano

Darine Abdullah



This work is licensed under a [Creative Commons Attribution-ShareAlike 4.0 International License](https://creativecommons.org/licenses/by-sa/4.0/).

Index

Introduction	2
Background.....	2
Solution.....	3
Creating a target data set	4
Processing data set.....	4
Cluster data set	4
Initialization	5
Assignment	5
Update	5
Stop criteria.....	6
Replace missed values.....	6
Export data set	6
Compare data sets	6
Algorithmic complexity	7
Description of tests	7
User's manual	8
Conclusions	10
References	11
Licence	12

Introduction

Clustering is a powerful machine learning tool that divides data set into a number of distinct groups based on some criteria. However, most clustering algorithms require having all the features of all the objects of the data set. This means that objects with missing values for one or more features cannot be clustered.

To address this problem, different approaches for handling data with missing values are proposed for different kinds of problems. [1]

- **Marginalization:** remove missing values.
 - **Feature marginalization:** omit features with missing values.
 - **Object marginalization:** omit objects with missing values.
- **Imputation:** attempts to “fill in” any missing values by inferring new values for them.
 - **Mean imputation:** replace each missing value with data set mean.
 - **Probabilistic imputation:** replace with random value according to data set distribution of values or using regression, etc.
 - **Data mining imputation:** determine the value using data mining techniques such as decision trees, clustering algorithms, etc.

Generally, missing values can occur in data sets in different forms: recording problem, instruments limitations, optional fields, etc. We could classify them into three cases: [2]

1. Missing values occur in several attributes (columns).
2. Missing values occur in a number of instances (rows).
3. Missing values occur randomly in attributes and instances.

Depending on the case we should use one method or another for dealing with the missing values.

The objective of this project is to develop algorithms to deal with missing values by using k-mean clustering. For large datasets with missing values, complicated methods are not suitable because of their high computation cost. We are trying to find simple methods that can reach performance as good as complicated ones.

Thus, we are not interested in marginalization approaches, we focus in imputation techniques, specially in data mining approaches.

Background

In the literature, imputation methods are classified into two groups according to when missing values are dealt: [2]

- **Pre-replacing methods:** replace missing values before the data mining process.
- **Embedded methods:** deal with missing values during the data mining process.

In pre-replacing methods we find:

Method	Cost	Attributes
Mean-and-mode method	Low	numeric / categorical
Linear regression	Low	numeric
Standard deviation method	Low	numeric
Nearest neighbor estimator	High	numeric / categorical
Decision tree imputation	Middle	categorical
Autoassociative neural network	High	numeric / categorical

Whereas in embedded methods:

Method	Cost	Attributes
Casewise deletion	Low	numeric / categorical
Lazy decision tree	High	numeric / categorical
Dynamic path generation	High	numeric / categorical
C4.5	Middle	numeric / categorical
Surrogate split	Middle	numeric / categorical

In data mining the complexity of the algorithm is extremely relevant because huge amounts of data are dealt. That is why, the methods that have “low” cost are the most used.

In this project, we are interested in a cluster-based simple enough algorithm that can deal with large datasets. We just want to fill the missed values, not to perform any other data mining process, so we focus in the pre-replacing methods.

The results showed in the previous table suggested us that mean method can be a good option.

Solution

The basic idea of our solution is the cluster-based filling up of missing values. After prepare the data set, we cluster it using the k-means algorithm without taking into account the missed values. Then, we calculate the mean of each feature of each cluster. And finally, we replace the missed values of each cluster by the mean of the feature.

In what follows we shall explain all steps in detail.

Creating a target data set

The first step is prepare the data set that we are going to use.

1. The data set must be in a CSV file.
2. The CSV must have the following format:
 - a. The values are separated by comma ','.
 - b. The first line is composed by the names of the features.
 - c. The second line is composed by the types of each feature. There are two types:
 - i: integer.
 - d: number with decimals (the separator for the decimal places is the point).**We don't support categorical features directly. We work in an n-dimensional Euclidean space. To use them, previously they have to be translated to numeric values. For example, if we have a feature which can have the values {A,B,C}, we translate them to {0,1,2}.*
 - d. The following lines are the records. One line per record.

And example of valid data set CSV file:

```
month,temperature,rain,      // Names of the features
i,d,i                        // Type of each feature
1,5.5,50                     // records 1 (point 1)
2,8.2,23                     // records 2 (point 2)
3,10.1,11                    // records 3 (point 3)
...                           // ...
```

Processing data set

The first thing the application does is to parse the CSV file. It reads the features, its types and all the records. While it is reading, it does several things:

- For each records, it creates a Point object where the values of features of the point are stored.
- It also detects missed features and registers the features and points which have them.
- At the same time, it calculates the mean and the standard deviation of each feature (to standardize the points later).

Cluster data set

We have the data set prepared, now we have to cluster the data set. But to do that, first we must standardize the values of all features, for them to have the same weight in the process.

To standardize a value of a point we have to subtract the mean of its feature and divide it by the standard deviation: $z = (x - \mu) / \sigma$

At this time, we are ready to apply k-means algorithm.

K-means is an iterative algorithm that keeps assigning data points to clusters identified by special points called centroids, until the cluster assignment stabilizes. [3] The number of clusters to create is determined by the parameter k, chosen by the user.

**The algorithm doesn't take into account the features which have missed values.*

The steps of k-means algorithm are the following:

Initialization

In the first steps k initial centroids need to be chosen.

Our method creates k initial cluster. The centroids's features are randomly chosen between the maximum and minimum value of each feature.

Assignment

The next step is to assign each point of the data set to the nearest cluster. Each point is assigned to the cluster which has the minimum Squared Euclidian Distance from the point to the centroid of the cluster.

The Euclidean distance between points p and q is the length of the line segment connecting them. In Euclidean n-space, then the distance (d) from p to q , or from q to p is given by the Pythagorean formula: [4]

$$\begin{aligned} d(\mathbf{p}, \mathbf{q}) &= d(\mathbf{q}, \mathbf{p}) = \sqrt{(q_1 - p_1)^2 + (q_2 - p_2)^2 + \cdots + (q_n - p_n)^2} \\ &= \sqrt{\sum_{i=1}^n (q_i - p_i)^2}. \end{aligned}$$

Since we only need to compare the relative distance between points, we can simplify the math and get rid of the square root. The Squared Euclidean distance place progressively greater weight on points that are farther apart. In this case, the equation becomes:

$$d^2(p, q) = (p_1 - q_1)^2 + (p_2 - q_2)^2 + \cdots + (p_i - q_i)^2 + \cdots + (p_n - q_n)^2.$$

Update

Once we have all the points assigned to a certain cluster, we update the centroids of all cluster. The new centroid of a cluster is the mean of all the points assigned to that cluster.

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

Stop criteria

The Assignment and Update steps are repeated until all new centroids are the same as the previous ones.

*As it is a heuristic algorithm, there is no guarantee that it will converge to the global optimum, and the result may depend on the initial clusters. As the algorithm is usually very fast, it is common to run it multiple times with different starting conditions. However, in the worst case, k-means can be very slow to converge: in particular it has been shown that there exist certain point sets, even in 2 dimensions, on which k-means takes exponential time, that is $2^{\Omega(n)}$, to converge. [5]

After the execution of the algorithm we get a collection of k clusters with their respective points. We must destandardize its values to get the originals: $x = \mu + z\sigma$

Replace missed values

Once we have the data set clusterized, we are ready to fill the missed values.

For each cluster that has missed values, we calculate the mean of the features that have some missed value (we don't take into account the points which have a missed value for the feature).

And finally, we replaced the missed values by the mean of the feature in the cluster. If the type of the feature is "integer", we round the value removing the decimal part.

Export data set

At last, the data set is exported to a CSV file in the folder chosen. The format is the same as the required format in the input data set.

Compare data sets

For testing purposes, our application has the ability to compare the original data set with the output data set, to show how many of the missed values have been filled correctly.

The application parses the original data set, the data set with missed values and the output data set. It uses the data set with missed values to register these missed points. Comparing the original with the output data set, it notices the error which has been committed. To have a relative measure of this error, the distance between the real value and the obtained value is divided by the distance of the range of values of the feature they belong.

Algorithmic complexity

The computational complexity of our application is determined by the k-means algorithm used to cluster the data set.

This algorithm is very fast in practice, we can cluster part of the US Census of 1990 data set which has more than 30.000 records and almost 70 features and it converges in less than 30 iterations. [6]

The complexity of k-means when k and d (dimensions) are fixed is: [5]

$$O(n^{dk+1} \log n)$$

where n is the number of entities to be clustered. But generally it is taken as: [7]

$$O(Iknd)$$

where I is the number of iterations.

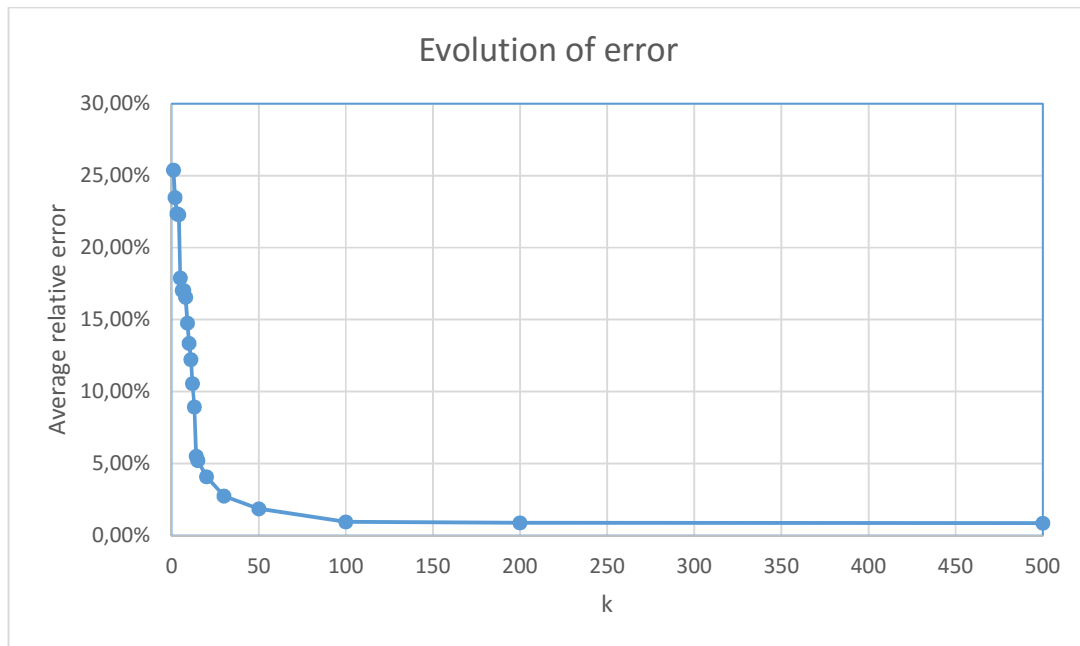
Description of tests

To test our application we took a complete data set and removed some values, then we filled the removed values with our application, and finally, we compare the obtained data set with the original.

The data set used has 1024 artificial records with 32 features each one of them. [8]

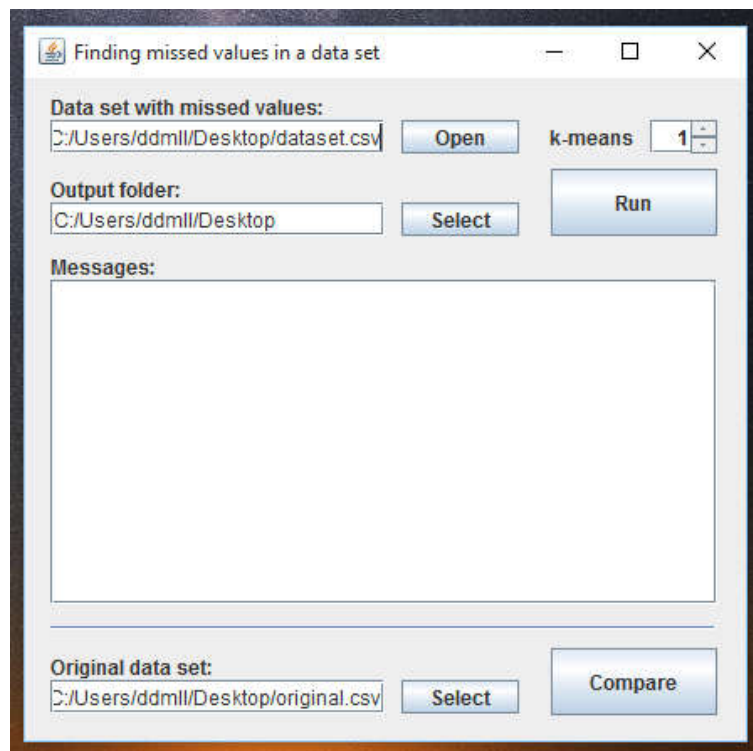
To measure the error of the obtained values, we compare the distance between them and the real values divided by the distance of the values range of the feature they belong.

In the next chart, we can see how the error decreases as the k increases.

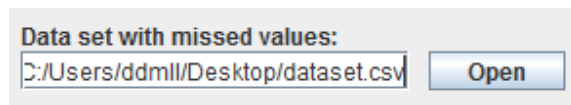


User's manual

The application is distributed as a executable JAR, when you run it, this screen is shown:

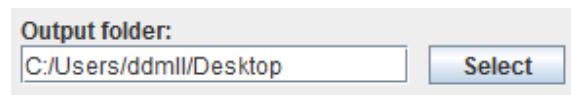


First, you have to select the data set with the missed values:



**The required format of the data set was explained in the previous sections.*

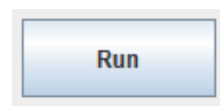
Then, you have to select the folder where the obtained data set will be saved:



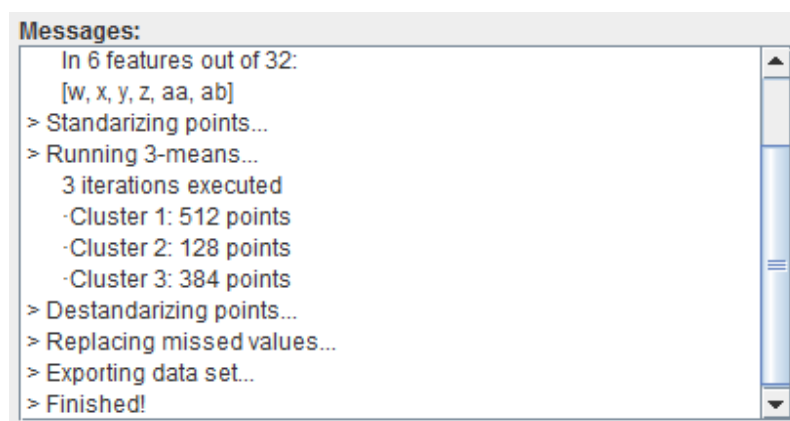
After that, you have to choose the k which will be used in the k-means algorithm:



Now, you are ready to run the application:



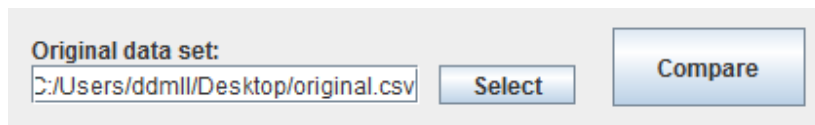
During the execution of the application, you can see the tasks which it is doing and the details in the console:



When the execution finish, the result data set is saved in the folder chosen.

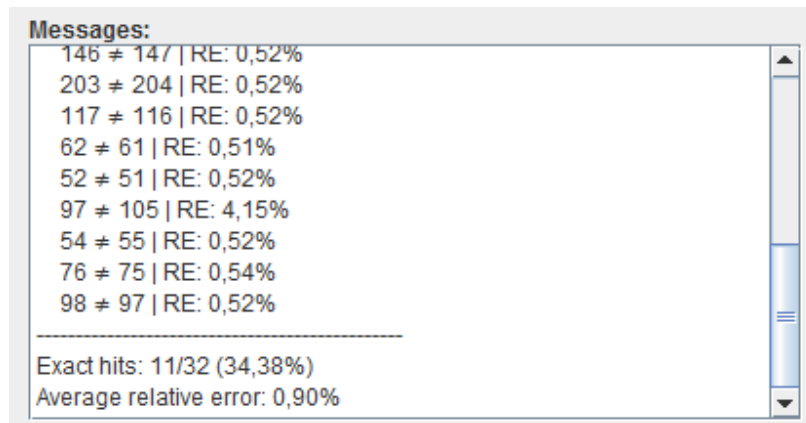
After that, you can compare that data set with the original data set to see how similar they are.

First you have to select the original data set, and then click on Compare button:



The screenshot shows a software interface with a text box labeled "Original data set:" containing the file path "C:/Users/ddmll/Desktop/original.csv". To the right of the text box are two buttons: "Select" and "Compare".

You will get the result in the console:



The screenshot shows a console window titled "Messages:". It displays a list of comparisons between original and inferred values, along with their relative errors (RE). At the bottom, it shows the total number of exact hits and the average relative error.

Original Value	Inferred Value	RE
146	147	0,52%
203	204	0,52%
117	116	0,52%
62	61	0,51%
52	51	0,52%
97	105	4,15%
54	55	0,52%
76	75	0,54%
98	97	0,52%

Exact hits: 11/32 (34,38%)
Average relative error: 0,90%

First, the wrong values are shown together with the real value and the relative error made. After them, you can see how many values have been inferred successfully, but the most relevant date is the average relative error which really does show how similar the new data set is.

Conclusions

In this project, we have demonstrated that it is possible to infer the missed values of a data set using clustering techniques. In particular, using the k-means algorithm, a simple and fast clustering algorithm. We have seen that the relevancy of choosing a proper k for the k-means and how this affects to the quality of the data set obtained.

A possible improvement in the application could be the implementation of a method to determine the optimal number of clusters (k). This can be achieved by using Bayesian Information Criterion (BIC).

References

- [1] Yoshikazu Fujikawa and TuBao Ho, «Cluster-based Algorithms for Filling Missing Values», Japan Advanced Institute of Science and TechnologyTatsunokuchi. Available at <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.453.7597&rep=rep1&type=pdf>
- [2] Kiri L. Wagstaff and Victoria G. Laidler, «Making the Most of Missing Values: Object Clustering with Partial Data in Astronomy», Astronomical Data Analysis Software and Systems XIVP2.1.25ASP Conference Serie. Available at <http://ml.jpl.nasa.gov/papers/wagstaff/wagstaff-missing-adass04.pdf>.
- [3] David Kosbie, «Introduction to Data Analysis: Clustering». Available at <http://www.kosbie.net/cmu/fall-10/15-110/handouts/notes-clustering/notes-clustering.html>.
- [4] Wikipedia contributors. «Euclidean distance». Wikipedia, The Free Encyclopedia; 2015 Sep 8, 07:11 UTC. Available at: https://en.wikipedia.org/wiki/Euclidean_distance.
- [5] Wikipedia contributors. «k-means clustering». Wikipedia, The Free Encyclopedia; 2015 Nov 26, 22:42 UTC. Available at: https://en.wikipedia.org/wiki/K-means_clustering.
- [6] University of California, «US Census Data (1990) Data Set». Available at <https://archive.ics.uci.edu/ml/datasets/US+Census+Data+%281990%29>.
- [7] Tan Steinbach and Kumar Ghosh, «The k-means algorithm». University of Vermont. Available at <http://www.cs.uvm.edu/~xwu/kdd/Slides/Kmeans-ICDM06.pdf>.
- [8] Speech and Image Processing Unit, «Clustering datasets». University of Eastern Finland. Available at <https://cs.joensuu.fi/sipu/datasets/dim032.txt>.

Licence

David Miguel Lozano
Darine Abdullah
Data Mining
POLITECHNIKA WARSZAWSKA
2015



This work is licensed under a [Creative Commons Attribution-ShareAlike 4.0 International License](https://creativecommons.org/licenses/by-sa/4.0/).

You are free to:

- **Share:** copy and redistribute the material in any medium or format
- **Adapt:** remix, transform, and build upon the material for any purpose, even commercially.