An Alternative Method to Preprocess Partly-Duplicated Data by Clustering Used for Machine Learning-Take A Black Friday Purchasing Record for Example

xx xx  
*xxx   
xx   
xx*  
chromeatease@outlook.com

*Abstract*—Deep Learning is one of the next big things in the intelligence system. The past few years have seen the tremendous success of deep neural networks in a number of complex machine learning tasks such as computer vision, natural language processing and speech recognition. Data is the most important component of Deep Learning, sometimes it can be very complicated to deal with. To figure out one of the problems, in this article, we proposed a method to preprocess the huge amount of data which are partly duplicated. We used an improved clustering algorithm based on K-Means combined with Jaccard Similarity to cluster and simplify the data. And then we build up a Deep Neural Network to made some comparisons between the training results of the preprocessed data and the raw data. Finally, we gave some possible ways to improve the method.

*Keywords—Deep Learning, Neural Network, Clustering, Data Preprocessing*

# Introduction

Analyzing, interpreting and making maximum use of the data is difficult and resource demanding due to the exponential growth of many business, governmental and scientific databases. It been estimated that the amount of data stored in the world’s database grows every twenty months at a rate of 100%. This fact shows that we are getting more and more exploded by data/information and yet ravenous for knowledge. Data mining therefore appears as a useful tool to address the need for sifting useful information such as hidden patterns from databases. In today’s world, where the accumulation of data is increasing in an alarming rate, understanding interesting patterns of data is an important issue to be considered to adjust strategies, to make maximum use of it, and find new opportunities. Organizations keeping data on their domain area takes every record as an opportunity in learning facts. However, the simple gathering of data is not enough to get maximum knowledge out of it. Thus, for an effective learning, data from many sources must first be gathered and organized in a consistent and useful manner. Data warehousing allows the enterprise to recognize what it has noticed about its domain area. The data must also be analyzed, understood, and turned into actionable information. This is the point where the application of data mining is needed [1].

With the rapid growth of data and the advances in computing power, data mining technologies have changed the world in many fields. One of the promising fields is Finance due to a wide variety of needs and the comprehensive data. For example, financial technology (FinTech) has fundamentally changed the applied marketing activities as the Internet increasingly becomes the leading advertising channel. Moreover, there are various electronic payment services available on mobile phones and smart credit cards, of which the records can serve as a medium for modeling customer behaviors. Therefore, not only e-commerce companies, such as Alibaba and Amazon, have expanded their business to provide financial services; but finance organizations including the banking sector are also eager to innovate new business models via user modeling [2].

Some connections between certain objects can be found by using data mining technologies such as clustering, deep learning, etc. With the huge development of information technology, more amount of data and more specific data can be collected by intelligent devices, apps and so on. These data are high correlated with users so that we can use them to analyze the user behavior to provide better and more suitable service to users. Customer behavior which is generated from customer’s purchasing records is a branch of the user data, these data can be used to do some goods recommendations, customers’ next purchasing prediction and so on.

However, sometimes huge amount of data are not very easy to deal with, especially within the lack of computing resources. Though, as for deep learning, the more data will be better, it can cause the *Curse of Dimensionality* and will have some negative effects to the result as if not to apply some certain preprocessing methods on it [9]. Every coin has two sides, at the same time you preprocess the data, some information will also lost for it. We can’t have it both ways, but we still can find some ways to try to preserve information as possible as we can as we preprocess them, meanwhile the scale of the data becomes smaller. We are not aimed to figure these kind of contradiction but to balance the performance and the accuracy.

Dataset which are not preprocessed can be very complex, there is a situation is that: The data can be partly-duplicated. To clarify these kind of situation, assuming we have a dataset that has 4 keys, which are, is generated by , while and are attributes of . In this case, can be duplicated and varied from different classes. In another words, is a set itself which has an uncertain length and a lot of but countable items. And our work is to convert the set to a numerical number which can indicate the most of the features of it.

We will give a general description about the dataset we have and also give a definition of the problem we are going to solve in Chapter 2. In Chapter 3, we will give a model and some algorithms. In Chapter 4, we will make a specific statement of the implementation and results.

# Definition of the Problem

## General Description of the Dataset

The dataset we have acquired is sponsored by the open source dataset project founded by *Kaggle.* It is a customer purchasing record set in Black Friday sales promotion, which contains 12 attributes, and they are *User\_ID,Product\_ID, Gender,Age,Occupation,City\_Category,Stay\_In\_Current\_City\_Years,Marital\_Status,Product\_Category\_1,Purchase*. Table. I shows the definition of each attribute.

1. Definition of each attribute in dataset

|  |  |
| --- | --- |
| **Attribute** | **Definition** |
| *User\_ID* | Index of each customer, started from the number 1 |
| *Product\_ID* | Index of a certain product, formatted like ‘P00069042’ |
| *Gender* | Gender of the customer, F refers to Female while M refers to Male |
| *Age* | Age of the customer, various from 0-17, 18-25, 26-35, 36-45, 46-50, 51-55, 55+ |
| *Occupation* | Occupation of the customer, various from 0 to 20 (integer) |
| *City*  *\_Category* | The district where the customer lives, various from A, B, C |
| *Stay\_*  *Current\_*  *City\_Years* | How long have the customer been in this city, various from 0, 1, 2, 3, 4, 4+ |
| *Marital*  *\_Status* | Whether the customer get married, 0 refers to single while 1 refers to married |
| *Product\_*  *Category\_1* | Which category the product belongs to(1-20) |
| *Purchase* | Price of the product |

## General Description of the Problem

The attributes in the dataset we’ve mentioned above are high correlated with each other, the attributes that belong to personal characteristics must have some connections with their purchasing attributes. From perspective of intuition, the elders would purchase more amount of products then the youngers, but as they getting much older, on the contrary, they would be more frugalness. In another case, the married individual’s purchasing behavior must be different from the singles’. Similarly, people with different occupations may also have some differences in purchasing behavior.

If you took a careful look at the dataset above, you may find out that not all the keys of it are the attributes of *User\_ID*. *Product\_ID* are generated by Users, it indicates the specific product that a user bought, and *Product\_Category*s are the attributes of *Product\_ID*, they represent the category that the corresponding product belongs to. As it shows in the Table I, there are 20 kinds of different categories of the products, and each user may has bought different kinds and number of the products.

We (only myself actually) define the dataset of similar patterns as *Partly-Duplicated Data*. Now we want to shrink the dataset to the form of “one user one row” for deep learning, and then the problem raises: Is there a way to shrink the data without losing a large amount of the patterns?

Abstractly speaking, given a dataset which each row has the form like:

Where , and rest of the items are numerical numbers.

Is there a way to generate its approximate set which has the form like:

Where all the items are numerical numbers?

# Model and the Algorithm

Take an insider look at each row, in which each item can be considered as the attribute of the row. As to say, every we’ve mentioned is some certain so called *behavior* of the corresponding row. In some recommendation systems, users are often classified to some different clusters according to their behaviors [3]. Similarly, we can also classify these rows according to the attribute . In this case, the numerical number represents the index of a certain class that the corresponding row belongs to. We will introduce some background knowledge first, and then we’ll make some modifications to them to fit our model. Finally, we’ll give the model and algorithm.

## The Jaccard Similarity Coefficient

Also known as the *Jaccard index*, and the *Intersection over Union*, and the *Jaccard similarity Coefficient* is a statistical measure of similarity between sample sets. For two sets, it is defined as the cardinality of their intersection divided by the cardinality of their union [4]. The *Jaccard Coefficient* measures similarity between finite sample sets, and is defined as the size of the intersection divided by the size of the union of the sample sets. Mathematically:

Especially, when :

## K-Means Clustering Algorithm

*K-Means* clustering (MacQueen, 1967) is a method commonly used to automatically partition a data set into groups. It proceeds by selecting initial cluster centers and then iteratively refining them as follows [14]:

1. Each instance is assigned to its closest cluster center.
2. Each cluster center is updated to be the mean of its constituent instances.

The algorithm converges when there is no further change in assignment of instances to clusters.

*Euclidean Distance* is mostly used to calculate the distances between instances and cluster centers, but in our case, the distance cannot be simply calculated by this method. To figure it out, we (only me actually) made some modifications on *Jaccard Similarity* and *K-Means* algorithm, as well as gave some essential definitions.

## The Modified Jaccard Similarity Coefficient(MJSC)

In our case, we propose to use the *Modified Jaccard Similarity Coefficient* instead of *Euclidean Distance* in *K-Means* clustering algorithm. Look back to our dataset, some attributes can be a set with uncertain length, and the most important staff is that there can be some duplicated values in sets, for example while they are all multisets (eg.1). The duplicated value like and its multiplicity which is three, they all counts actually.

There are some other disadvantages for *Jaccard Similarity* for its using in this case, but we won’t mention here. We’ll give the definition of *Jaccard Similarity Coefficient* for multisets directly:

Where:

and are multisets,

is the number of items with distinct values in both and ,

are items of the set which contains distinct values in both and ,

are the multiplicity of in or .

As we defined, the *Modified Jaccard Similarity Coefficient* of eg.1 is:

## The Extended Barycenter of Multisets for K-Means Using

Recalculate the barycenter is one of the most important steps in *K-Means* algorithm, for that we still need to define the barycenter of the cluster of multisets. As we all acknowledged, in geometry, barycenter is the center of mass of two or more bodies that orbit each other and is the point about which the bodies orbit.

Barycenter is always connect with mass, similarly in our case, the value which is of lager multiplicity has the lager mass. So, follow the definition of barycenter in geometry, we gave the definition of its counterpart in multiset directly:

Where:

is the number of multisets in a cluster,

is a multiset which contains all kinds and amount of values in the multisets,

means there are duplications of in the corresponding multiset.

As we defined, the barycenter of the two multisets in eg.1 is:

## The Optimized K-Means Aglorithm Based on Modified Jaccard Similarity Coefficient

So long (finally… I’m already tired out) we have satisfied all the conditions to apply *K-Means* on a certain dataset, and then we are going to work on *K-Means* again.

Traditional *K-Means* clustering algorithm usually start with cluster centers which are randomly chosen. Based on the experiences of other researchers’ (they are dazhutizi), *K-Means* is *Initial Value Sensitive.* As to say, the clustering results can be very different in the case of different initial values. To fix the problem in a best effort, a method came to me (I borrowed and modified it actually, from Baidu).

As for clustering, we finally want the clusters to be far away from each other, that’s we called a pretty clustering result. So think about it, why don’t we select cluster centers which are already very far from each other at the very beginning? In our case, the work is: Choose multisets which have very small *Modified Jaccard Similarity Coefficient* with each other as initial cluster centers.

We will give pseudocodes for the initialization of cluster centers, and for the iteration of modified *K-Means* in Algorithm 1 and Algorithm 2. The whole and executable source codes can be found in my Zipped file.

|  |
| --- |
| **Algorithm 1:** Initialization of Cluster Centers |
| 01: random choose a multiset from whole multisets  02: add to the initial centers list , and calculate the **barycenter** of all the multisets in as  03: calculate the **MJSC** between and all other multiset in the whole multisets  04: find the smallest **MJSC** and its corresponding multiset, and then add it to  05: if the number of centers in is less than the amount of clusters , calculate the barycenter of all the multisets in , update and go to **03**, otherwise, return and exit |

|  |
| --- |
| **Algorithm 2:** Iteration of Modified K-Means |
| 01: copy to cluster centers  02: for each multiset in multisets, calculate the **MJSC** between itself and every cluster center in  03: find the smallest **MJSC** and its corresponding cluster and add **current multiset** to  04: if cluster don’t change anymore, then return and exit, otherwise, recalculate the **barycenters** of and update , then go to **02** |

# Implementation and Results

Eventually, we are in the chapter of Implementation, and there is still one chapter to go. So, cheer it up, we are almost there.

The dataset that we were tested on is the dataset we’ve mentioned in chapter 2, for that we are not going to describe it redundantly. In this chapter, we will test the algorithm firstly to see how it works, and evaluate the results of clustering by intuition. After that, we built a Neural Network and train it to predict the age of each user base on the different datasets, one of which is the data preprocessed by the algorithm, and the other, not. A comparison will be made to see how much performance we gain and how much accuracy we lost in the meantime.

## Facts of the Algorithm Gained After Testing

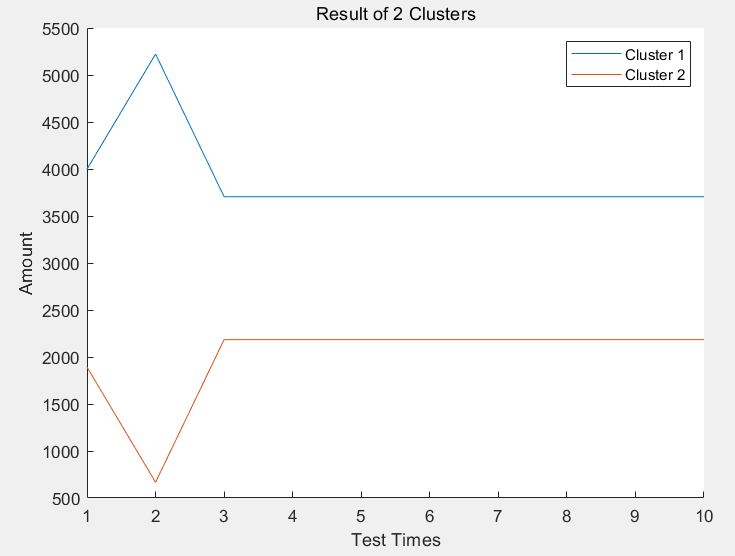
In our dataset, we can easily find out that the key named *Product\_Category\_1* of which the values are multiset, so intuition drives me to cluster the users in some certain classes according to it. After we applied the algorithm on it, we got some facts.

### The Algorithm Can be Hard to Converge in Some Cases: The algorithm will converge very fast in the case of small value, and as the value of continues to grow, the number of iterations grows. When it grows to a certain value, the result of clustering begins to vibrate with a certain range(very small but won’t converge). Table II shows the relationship bewteen value and average number of iterations.

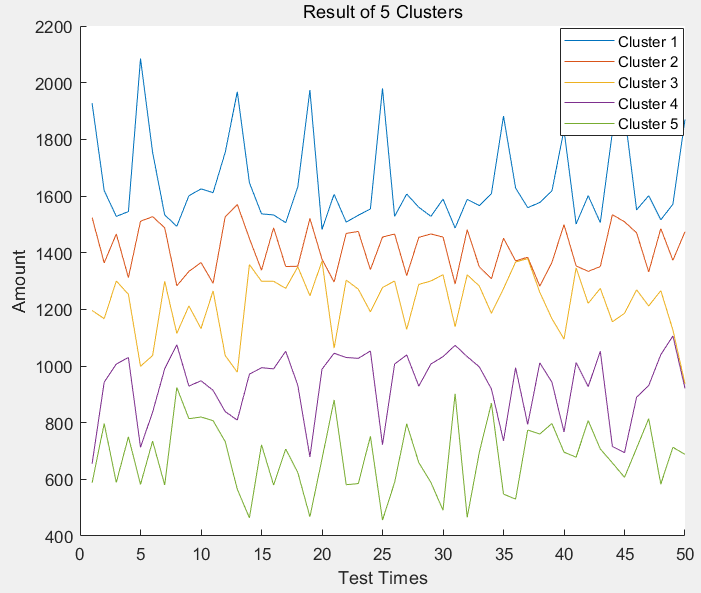
1. Value and Iteration Times

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ***Value*** | ***2*** | ***3*** | ***4*** | ***5*** | ***6*** | ***7*** | ***8*** | ***9*** | ***10*** |
| *Iteration times* | 10 | 15 | 22 | 35 | 48 | 62 | 77 | 123 | ∞ |

### Clustering Results Will Be Differ From Each Time: The clustering reasults will be different as if choose the different initial cluster centers, but interestingly, when equals 2, the result seems to be very stable. Figure I shows the results of clustering to different amount of clusters.



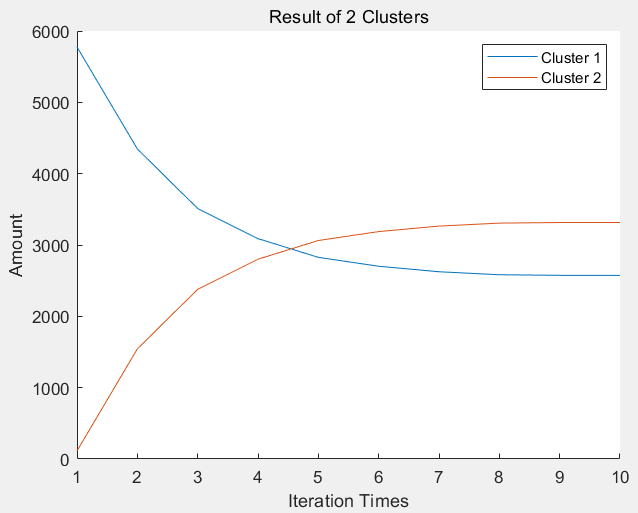
1. The Clustering Result of Two Clusters



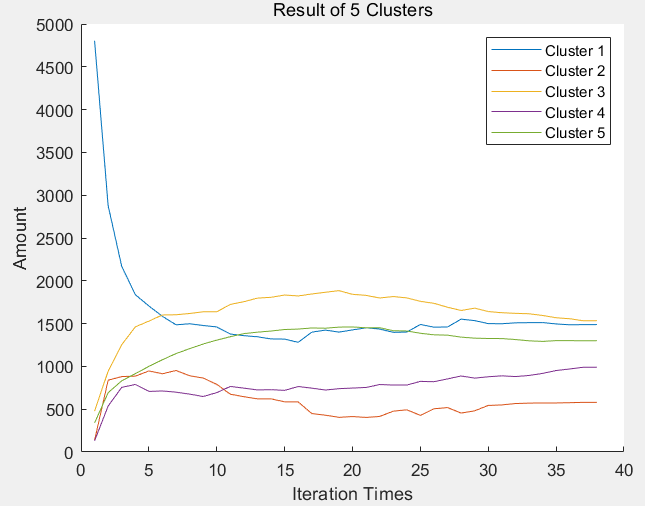
1. The Clustering Result of Five Clusters

Figure I. Results of Clustering to Different Amount of Clusters

### The result of the First Iteration and the Last Can be Very Different:The result of first iteration and the last is always very different, and it will change gradually till converge. Through figure II, we can acknowledge the process of convergence.



1. Instance of Two Clusters



1. Instance of Five Clusters

Figure II. Changes of the Amount of Each Cluster Alone with Iteration Times

Fortunately, the algorithm would finally converge ☺, no matter how it converges, let’s build the neural network and train it!

## Result of the Trainning of the Neural Network

We built two neural networks, one with nine dense layers and two dropout layers and the other has one dense layer and dropout layer. To be stressed, we used two extra datasets to make comparisons, one of which is the raw data which been slightly preprocessed like regulation, and the other dataset is the preprocesses dataset without the column which contains the clustering results.

All computing works are accomplished by my laptop with Intel i7-8550U quad-core processor @3.4Ghz, 16GB of memory running at DDR3L 1866Mhz. Software ran under Windows 10 professional 1809, and neural networks are built through TensorEditor, runs on tersorflow ver1.12.0 with python ver3.6. Table III gives the parameters and training results of the neural networks.

1. Parameters and Training Results of The Neural Networks

|  |  |  |
| --- | --- | --- |
| ***Name*** | ***NN1*** | ***NN2*** |
| ***Dense Layer*** | 9 | 2 |
| ***Dropout Layer*** | 2 | 1 |
| ***Learning Rate*** | 0.0001 | 0.001 |
| ***Batch Size*** | 100 | 100 |
| ***Shuffle Times*** | 1000 | 1000 |
| ***Rule*** | relu | relu |
| ***Steps of Train*** | 300000 | 10000 |
| ***Accuracy ①***  ***(******Preprocessed Dataset)***  ***5890 items*** | **32.56%** | **31.70%** |
| ***Accuracy ②***  ***(Raw Dataset)***  ***537577 items*** | **40.48%** | **24.25%** |
| ***Accuracy ③***  ***(Preprocessed Dataset***  ***Without Clustering Results)***  ***5890 items*** | **N/A** | **18.1%** |
| ***Time Costs (Average)*** | **8 h** | **3 min** |

The statistics showed in the Table III is in line with the intuition. The preprocessed dataset (①) is much easier to train, for the clustering algorithm has already find out some correlations in them. On the contrary, to training the raw data (②), more computing resources and time as well as the RAM space are needed, and only in this way, you can get a better result, for it can dig more hidden correlations from it. As for the dataset (③), proves that the clustering results make sense in a certain way.

# Conclusions

There still have a lot of work to do to optimize the model and the algorithm, but in this stage, it is still an alternative way to shrink dataset.

##### References

1. Z. Zamani, M. Pourmand, and M. H. Saraee, “Application of data mining in traffic management: Case of city of Isfahan,” in 2010 2nd International Conference on Electronic Computer Technology, 2010, pp. 102–106.
2. Y.-T. Wen, P.-W. Yeh, T.-H. Tsai, W.-C. Peng, and H.-H. Shuai, “Customer Purchase Behavior Prediction from Payment Datasets,” in Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining, New York, NY, USA, 2018, pp. 628–636.
3. P. Covington, J. Adams, and E. Sargin, “Deep Neural Networks for YouTube Recommendations,” in Proceedings of the 10th ACM Conference on Recommender Systems, New York, NY, USA, 2016, pp. 191–198.
4. J. Bank, “Calculating the Jaccard Similarity Coeﬃcient with Map Reduce for Entity Pairs in Wikipedia,” p. 18.R. Nicole, “Title of paper with only first word capitalized,” J. Name Stand. Abbrev., in press.
5. L. Backstrom and J. Leskovec, “Supervised random walks: predicting and recommending links in social networks,” presented at the Proceedings of the fourth ACM international conference on Web search and data mining, 2011, pp. 635–644. Yorozu, M. Hirano, K. Oka, and Y. Tagawa, “Electron spectroscopy studies on magneto-optical media and plastic substrate interface,” IEEE Transl. J. Magn. Japan, vol. 2, pp. 740–741, August 1987 [Digests 9th Annual Conf. Magnetics Japan, p. 301, 1982].
6. C. A. Gomez-Uribe and N. Hunt, “The Netflix Recommender System: Algorithms, Business Value, and Innovation,” ACM Trans. Manage. Inf. Syst., vol. 6, no. 4, pp. 13:1–13:19, Dec. 2015.
7. H. Guo, R. Tang, Y. Ye, and X. He, “Holistic Neural Network for CTR Prediction,” in Proceedings of the 26th International Conference on World Wide Web Companion, Republic and Canton of Geneva, Switzerland, 2017, pp. 787–788.
8. C. Kamada, A. Kanezaki, and T. Harada, “Probabilistic Semi-Canonical Correlation Analysis,” in Proceedings of the 23rd ACM International Conference on Multimedia, New York, NY, USA, 2015, pp. 1131–1134.
9. A. Karatzoglou and B. Hidasi, “Deep Learning for Recommender Systems,” in Proceedings of the Eleventh ACM Conference on Recommender Systems, New York, NY, USA, 2017, pp. 396–397.
10. M. Moricz, Y. Dosbayev, and M. Berlyant, “PYMK: friend recommendation at myspace,” presented at the Proceedings of the 2010 ACM SIGMOD International Conference on Management of data, 2010, pp. 999–1002.
11. P. Pecher, M. Hunter, and R. Fujimoto, “Data-Driven Vehicle Trajectory Prediction,” in Proceedings of the 2016 ACM SIGSIM Conference on Principles of Advanced Discrete Simulation, New York, NY, USA, 2016, pp. 13–22.
12. H. Zhang, W. Liu, and Z.-J. Zha, “Sparse Canonical Correlation Analysis for Recognition,” in Proceedings of the 7th International Conference on Internet Multimedia Computing and Service, New York, NY, USA, 2015, pp. 17:1–17:4.
13. Y. Zhang and Z.-H. Zhou, “Multilabel Dimensionality Reduction via Dependence Maximization,” ACM Trans. Knowl. Discov. Data, vol. 4, no. 3, pp. 14:1–14:21, Oct. 2010.
14. K. Wagstaﬀ, C. Cardie, S. Rogers, and S. Schroedl, “Constrained K-means Clustering with Background Knowledge,” p. 8.