Big Data without Map Reduce

Yanan Xiao, *Student Member, IEEE,* Aziza Al Sawafi

***Abstract*—We present our approach to the big data problem put forward in KDD Cup 2009. We carry out a more technical analysis of our initial plan. And we describe our trials and errors with the large dataset. Till now, our team has not come up with any practical solution with large dataset in a limited time. We present our results with the small dataset in the end.**

***Keywords*—Customer relationship prediction, ensemble selection, classification, data mining**

I. Introduction

CUSTOMER relationship management (CRM) is an essential model for managing the interactions between the company and its current and future customers. It takes up-to-date technologies to organize, automate, and synchronize CRM and marketing information system [1]. CRM applications that use data mining are called Analytic CRM, which provides valid predictions from customer data collected and stored with various attributes. There are a lot of data mining tools and methods to extract and analyze data generally, and customer data specifically. A naive step in doing so is to summarize the statistical attributes of the data (such as means and standard deviations) and use charts and graphs to review it visually [2]. However, in many situations customer relationship data volume is vast and massive. Therefore, more sophisticated methods are created and evaluated. In this field, we found the following techniques are most frequently employed: decision trees, support vector machines, artificial neural networks and Bayesian classifiers.

Data mining has powerful capability in processing and analyzing data; its key technologies that applied in CRM are categorized into three main categories; clustering, classification and forecast, and association rules [3]. Classification and forecast analysis classifies unknown data into the most proper predefined class based on category description that is obtained by training a set of data using certain algorithm. Key classification techniques are; decision making tree, Bayesian statistics, BP neural networks, Genetic Algorithm, rough set theory, fuzzy set theory and so on. Classification methods in CRM can predict new customers behaviors and activities.

As mentioned in [4], classification analysis is the one that is widely used in classifying CRM data. It can be processed in two steps; learning phase and training phase [5]. In the learning phase the classification algorithm analyzes the training data set and learns it, then in the second phase the accuracy of the classifier will be estimated using the test data set. After that, the classifier can be used to predict and classify new data set. In order to obtain better accuracy, some preprocessing and filtering techniques can be applied to the data before going through the classification phases. Those techniques are; data cleaning, data discretization, and feature selection. The common challenges in knowledge discovery from CRM data are the high-dimensionality, and imbalanced corrupted records. Besides, researchers roved that customer classification and prediction is cost sensitive in nature. For example, if a valuable customer predicted as loyal but then that customer churns, the cost is higher than if a loyal customer is classified as one who will churn [6].

Y. Xiao, A. Sawafi are 1st year master students with the Department of Electrical Engineering and Computer Science, Masdar Institute of Science and Technology, Masdar City, Abu Dhabi, UAE. P.O. 54224 Email: {yxiao,aalsawafi@masdar.ac.ae}.

One team member quit Masdar Institute not soon after the project. But we managed to get things done by ourselves.

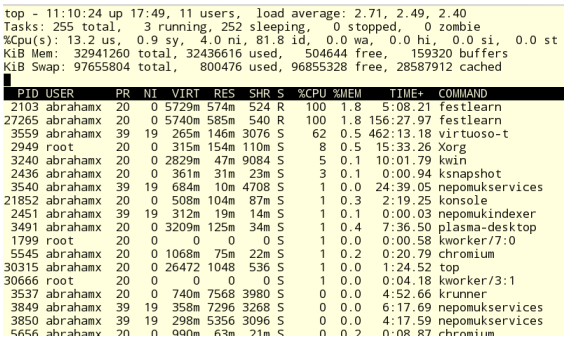

Fig. 1. CPU and memory usage when running FEST package

In this paper, we present various tools that we tried and several plans that were carried out when dealing with big data problem like this. The rest of the paper is organized as follows. Section II displays what our thoughts were before tackling the real large dataset. Section III shows what our efforts are when no team member has the pre-knowledge of how to write *map reduce* scripts and run them with the help of clusters. Section IV serves as the conclusion for this not-so-successful trial with big, complex data mining task.

II. Initial Proposal

Lesson we learn from writing and reviewing this section is that "think twice before taking actions". We depicted a master plan in our previous proposal [7]. However, due to the lack of map reduce technique, we are able to train models on a oct-core workstation. The problem is that the training part of ensemble selection is really memory consuming. All of the 32GB memory got used instantly while at the same time we observe the CPU usage just above 10%. See Fig 1.

A. Preprocessing

In order to tackle the challenges of the big data set, we propose the following steps.

Missing value. We would consider missing categorical values as a separate value. We would take a standard approach which calculates the mean of the feature to impute missing values. And as proven an effective technique by [8]. We decide to add an extra indicator variable to indicate missingness for every one of the 333 variables with missing values. We planned to do this because some linear models in our base classifiers could then estimate the optimal constant instead of merely relying on the means to replace the missing value with.

Categorical value. Since categorical values are not easily handled by many learning algorithms, we decide to recode categorical values using the same way as by IBM Research. For different values a categorical attribute could take, we would generate corresponding indicators. As a good example shown in IBMs paper, limiting the number of values encoded would greatly reduce the number of features, which may be from variables with an enormous vocabulary.

Clean up. We would normalize each feature by dividing up by their range. And we would clean the data by eliminating redundant features, which are either constant, or duplicate of other features.

B. Base Classifiers

As stated by Dietterich [9], "A necessary and sufficient condition for an ensemble of classifiers to be more accurate than any of its individual members is if the classifiers are accurate and diverse". Building libraries would be expensive, whats more is that time is limited to train all of them in the next phase. To maintain the accuracy and diversity of our next-phase base classifier library, we decide to build it using the classifiers as follows, boosted decision trees, decision trees, linear regression, SVM and k-NN.

C. Ensemble Selection

An ensemble is a collection of models. To make predictions, we just need to calculate weighted average or "voting" based on those models. One important reason for us to work with ensemble selection is that it can be optimized to any easily computed metric. Same as described before, for an ensemble selection classifier used to improve classification accuracy, one thing is the base classifiers in the library are accurate. This is not hard to achieve with a great literature at hand. The other characteristic is diverse. Due to the fact that our computation power is limited, we planned to build 100 to 200 base classifiers in our library.

D. More Features and Base Classifiers

We planned to create some more features and base classifiers. One is to buildup a decision tree using FEST package [10], and then visualize this tree so that we can tell distinguish a set of nodes forming the best splitting points. We also refer to a few state-of-the-art classifiers that claiming better performance than traditional ones. Nevertheless, as put out in the following section, we did not possess enough energy to handle them.

III. Our Journey

As mentioned in abstract, we are not able to produce result which exactly follows what we depict in our midterm report. Our initiative is to follow the step of IBM Research, which outperformed other teams in KDD Cup 2009. We started our more detailed background investigation right after the submission of midterm report. Our first attempt was to set up a cluster with the help of Amazon Web Service[1]. We tried several times but were not able to make the whole cluster share a same virtual disk, which is vital according to [11]. Our review on this failure is that we should set off even earlier after being informed that this is a real classification problem on really complex data. After the submission, we also investigate tools like hadoop, MongoDB and Apache Mahout[12]. Because the time is limited and each team member has other research task at hand, we plan to run our initial plan on a Amazon cluster later on. Below we discuss our process step by step.

A. MATLAB

Our preprocessing is done fairly standard and systematic. We use MATLAB to load the plaintext feature chunk and then run pre-written scripts to format them following the previous guidelines. A note here is that when dealing with large datasets like this, it's also memory consuming. The memory consumption in total for train and test chunk sums up to 10GB. We then use the built-in function *csvwrite* to format them. This is important because tools we use, like FEST package, BBR package, and LIBSVM, only supports LIBSVM format. The final conversion is done with a c program that helps transform csv format files into libsvm format. This step is also done automatically by writing a shell script.

One thing that is worth noting here is that we actually did two types of preprocessing, one pretty naive whereas the other strictly following the standard. So for our team we have as many as two sets of large data and two sets of small data. Our story with small data is discussed in the next section since we finished it after the deadline, because we really want to produce some results. For the naive preprocessing, we merely transformed the categorical data to numerical, and skip any other step. We later find this strategy performed badly due to some obvious reasons: large dimensionality, large data scale for transformed categorical data, and redundant or even noisy feature.

After the first attempt of "lazy" preprocessing, the strict version is carried out as follows:

1) Impute both missing numerical and categorical values with their mean at that column.
   - If one column is composed of all NaN values, it is tagged 1.
   - Else it is tagged 2.
   - The value that is tagged 1 then gets deleted because no information could be obtained from that feature. This treatment is also useful to indicate missingness.

[1] http://aws.amazon.com/ec2/

2) Recode the categorical values by sorting and selecting. We select top 10 largest categorical events, and recode them by computing their frequency in the corresponding feature. If there are fewer than 10 events, all of them get kept and recoded.
3) The last step we do is to normalize each feature value and then mapping them to scale $[-1, 1]$.

We also plan to carry out some classification with MATLAB built-in function *TreeBagger* later on. Since both team member agree that following IBM's method is a good way, for the large dataset we still try to tackle it with FEST packages and the alike.

B. Weka

Weka is the first integrated tool that came to our mind. The author mentioned in the manual that even a sample of real-world data that's processed with ensemble selection could take days or even weeks to run[11]. There are two built-in model libraries in weka's ensemble selection package, both are "incredibly" large. We follow the instructions on this manual and describe our attempt with Weka as follows:

1) Build a model list with library editor GUI implemented in weka's ensemble selection package. One thing worth noting here is that since the newest stable version, EnsembleSelection package is not shipped as a built-in package. The model list file is actually a xml file containing command information to be read and executed by weka.
2) The file we try with weka is in csv format. It will be transformed to arff format once the classification, in our case ensemble selection command, is entered. Part of our command is supplied in the appendices.

As indicated in manual "Ensemble Selection in a Nutshell", training takes time when the dimension of data is large. And this is especially the case with KDD Cup's dataset. We deal with more than 14,000 features this time because we did not delete as many features as IBM did. Moreover, the large dataset contains 50,000 instances. We did run weka to get a result successfully. Training that feature requires almost 24 hours and one model file output by the Ensemble Selection weka implementation would take as much as 4.2GB harddisk space. Since this is the first time that we deal with *big data* using weka, with no help from tools like hadoop to map reduce the problem, we decided to abandon it for practical considerations, like time, harddisk space, etc.

C. Specific Packages

After the first two trials, we find that it is too resource consuming to train and classify big data using ensemble selection method via more generic tools like MATLAB and Weka. And that seem to explain why IBM research team did not employ tools as such. Another interesting thing we find is that even software like Hadoop that is built to help distributed computing is written in Java, which appears to be the "best practice" for some time.

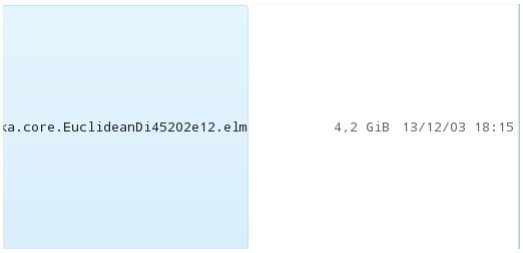

Fig. 2. One model file trained by Weka ensemble selection package

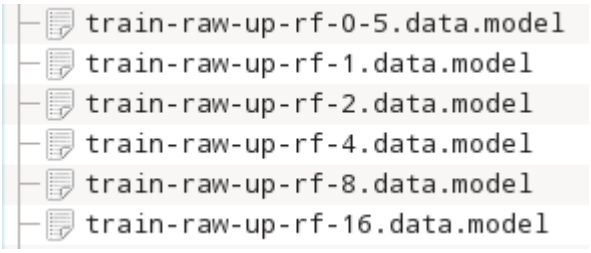

Fig. 3. Model files trained by FEST package

Fig. 4. Corresponding training time for some random forest models

We choose FEST, LIBSVM and BBR packages in the end. For FEST package we use it to train random forest as well as boosted decision trees. Even with the fully preprocessed data, it still takes a long time to train *one* random forest or boosted decision tree. In Figure 3 it is demonstrated how time consuming it is when the package program was not written for parallel purpose. And Figure 4 shows the related training time for those models.

The LIBSVM package [13] is used to train models based on support vector machine technique. We also prepared shell scripts for this step. Unfortunately, due to the practice in map reduce technique, even if the CPU of our workstation has a 80% idle rate, we can hardly run all those scripts in parallel. The BRR package is obtained for logistic regression usage for which we also prepared automatic shell scripts [14].

Without the assistance from powerful map reduce tools, till now we have trained random forest models of appetency, churn and upselling. The classification is still on-going. We also manually run the *festclassify* program during the training process, the best result we get is 0.991667 as the prediction probability for one instances. The other prediction file that we have has the distribution as shown in Figure 5. The distribution looks fine but we haven't found out any reasonable explanation why the mean of this prediction is only 0.1034.

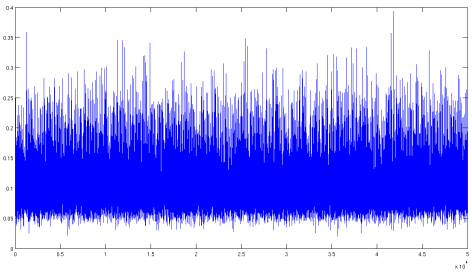

Fig. 5. Prediction Distribution for One Churn Model

For the 0.991667 prediction, we suppose that overfitting is one possible explanation.

D. Small Dataset

In addition to our original plan, we also practiced with the small dataset the moment we find it impractical to stick with ensemble selection. Overall ensemble selection as described in [10] can have good performance on high dimension dataset, but that point stands on the basis of carefully running either Weka program or specific packages *in parallel*. Due to the lack of related knowledge in map reduce, we run similar classifiers on Weka with small dataset. We describe our results as follows.

The preprocessing is carried out in a similar manner not parameters are not 100% the same.

1) If an feature contains more than 50% missing values, i.e. "NaN" in MATLAB terminology, then this feature is no longer considered.
2) For the remaining features, we impute missing values as follows:
   a) Both missing values in numerical and categorical features are imputed with their modes and means.
   b) Categorical values are recoded to numerical values, by their relative frequency.
   c) For each feature the discretization is applied to smooth distribution curve.
3) The final clean up is done by filtering features that possesses a less than 0.01 information gain. We use Weka's built-in function InfoGainAttributeEval filter to achieve this step.

For small dataset, we also carry out the classification with two samples of data. One set we removed categorical features completely the other is full, for the benefit of comparison. The base classifiers we use in this case are Naive Bayesian Classifier (NBC), Lazy IBK and Decision Table Classifier (DTC). NBC is simple in structure and it is based on the assumption that predictors are conditionally independent given the target variable. Because of its simplicity, NBC was attractive choice since we have a large set of variables. The second classifier is Lazy IBK (Instance-Based K) which is one of the Nearest Neighbor algorithms. Thirdly, we used DTC which uses a simple decision table majority algorithm; it makes decisions on attributes for each instance.

Our results with the small dataset is displayed as follows.

TABLE I. SMALL DATASET WITHOUT CATEGORICAL VALUES

Method	AUC			
	Churn	Appetency	Upselling	Score
NBC	0.529	0.497	0.658	0.561
Lazy IBK	0.532	0.525	0.619	0.558
DTC	0.53	0.5	0.757	0.595

TABLE II. SMALL DATASET WITH FULL PREPROCESSED VALUES

Method	AUC			
	Churn	Appetency	Upselling	Score
NBC	0.568	0.523	0.682	0.591
Lazy IBK	0.533	0.5	0.564	0.532
DTC	0.53	0.5	0.756	0.595

IV. Results and Conclusion

For the implementation of our blueprint that was discussed in our midterm report, we, as a solid team of two, have tried our best to obtain results using ensemble selection technique. Our results with large dataset are far from being satisfactory and we plan to have some discussions with course lecturer to figure out possible reasons.

We set off at an early time for this final project, but neither team member has sufficient technical background in this field to solve this project "safe and sound". We are a team of two but each other did the following things:

- Yanan Xiao. He tried various methods to tackle large dataset and wanted to stick to the team's initial plan, i.e. ensemble selection with around 300 base classifiers. He is in charge of the report typesetting in IEEE template with LaTeX2e.
- Aziza Al Sawafi. She came up with the small dataset workaround and successfully got some results. She also tried to solve this classification problem using MATLAB and LIBSVM.

Appendix A
Source Code

We enlist our partial source codes here. The complete project is at Github CIS501 repository[2].

A. Preprocessing

```
%-----Recode--------%
Recode={};
for i=14703:14962
    value=unique(Train(:,i));
    value(value==0)=[];
    if ~isempty(value)
    value(:,2)=0;
    for j=1:size(value,1)
        value(j,2)=sum(Train(:,i)==
        value(j,1))/50000;
    end
    [~,lev]=sort(value(:,2),'descend');
    value=value(lev,:);
    if length(value)>10
```

[2] https://github.com/ProfessorX/CIS501

```
    value=value(1:10,:);
    end
    end
    Recode{1,i-14702}=value;
    value1=unique(Test(:,i));
    value1(value1==0)=[];
    if ~isempty(value1)
    value1(:,2)=0;
    for j=1:size(value1,1)
        value1(j,2)=sum(Test(:,i)==
        value1(j,1))/50000;
    end
    [~,lev]=sort(value1(:,2),'descend');
    value1=value1(lev,:);
    if length(value1)>10
    value1=value1(1:10,:);
    end
    end
    Recode{2,i-14702}=value1;
end
```

B. Automatic Shell Scripts for Building Models

```
#!/bin/bash
festlearn -c 3 -n 0.1 -p 0.5 -t 300
train-raw-ap.data
train-raw-ap-rf-0-5.data.model &&
festlearn -c 3 -n 0.1 -p 1 -t 300
train-raw-ap.data
train-raw-ap-rf-1.data.model &&
festlearn -c 3 -n 0.1 -p 2 -t 300
train-raw-ap.data
train-raw-ap-rf-2.data.model &&
festlearn -c 3 -n 0.1 -p 4 -t 300
train-raw-ap.data
train-raw-ap-rf-4.data.model &&
festlearn -c 3 -n 0.1 -p 8 -t 300
train-raw-ap.data
train-raw-ap-rf-8.data.model &&
festlearn -c 3 -n 0.1 -p 16 -t 300
train-raw-ap.data
train-raw-ap-rf-16.data.model &&
festlearn -c 3 -n 0.1 -p 32 -t 300
train-raw-ap.data
train-raw-ap-rf-32.data.model &&
festlearn -c 3 -n 0.1 -p 64 -t 300
train-raw-ap.data
train-raw-ap-rf-64.data.model &&
```

ACKNOWLEDGMENT

The authors would like to thank Dr. Wei Lee for giving high quality data mining lectures and selecting this challenging but rewarding topic as this semester's project. They would like give more gratitude to Masdar Institute for the studying and researching environment.

REFERENCES

[1] Wikipedia. (2013) Customer relationship management. [Online]. Available: https://en.wikipedia.org/wiki/Customer_relationship_management

[2] A. Al-Mudimigh, Z. Ullah, and F. Saleem, "Data mining strategies and techniques for crm systems," in *IEEE International Conference on System of Systems Engineering*, 2009, pp. 1–5.

[3] K. Wu and F. Liu, "Application of data mining in customer relationship management," in *Management and Service Science (MASS), 2010 International Conference on*, 2010, pp. 1–4.

[4] N. Shahrokhi, R. Dehzad, and S. Sahami, "Targeting customers with data mining techniques: Classification," in *User Science and Engineering (i-USEr), 2011 International Conference on*, 2011, pp. 212–215.

[5] I. Guyon, V. Lemaire, M. Boulle, G. Dror, and D. Vogel, "Analysis of the kdd cup 2009: Fast scoring on a large orange customer database," in *Proceedings of KDD-Cup 2009 competition*, Paris, France, Jun. 2009, pp. 23–34.

[6] M. Lobur, Y. Stekh, and V. Artsibasov, "Challenges in knowledge discovery and data mining in datasets," in *Proceedings of VIIth International Conference on Perspective Technologies and Methods in MEMS Design (MEMSTECH)*, 2011, pp. 232–233.

[7] Y. Xiao and A. Sawafi, "Getting the max out of ensemble selection," *Journal of Masdar Institute*, vol. 1, pp. 323–330, Dec. 2013.

[8] A. Niculescu-Mizil *et al.*, "Winning the KDD Cup orange challenge with ensemble selection," in *Proceedings of KDD-Cup 2009 competition*, Paris, France, Jun. 2009, pp. 23–34.

[9] T. G. Dietterich, "Ensemble methods in machine learning," in *Multiple Classifier Systems*, J. Kittler and F. Roli, Eds. New York: Springer Verlag, 2000.

[10] R. Caruana, N. Karampatziakis, and A. Yessenalina, "An empirical evaluation of supervised learning in high dimensions," in *Proceedings of the 25th International Conference on Machine Learning*, ser. ICML '08. New York, NY, USA: ACM, 2008, pp. 96–103. [Online]. Available: http://doi.acm.org/10.1145/1390156.1390169

[11] *Ensemble Selection in a Nutshell*, Online Manual, Weka Wiki, 2009.

[12] C.-T. Chu, S. K. Kim, Y.-A. Lin, Y. Yu, G. Bradski, A. Y. Ng, and K. Olukotun, "Map-reduce ofr machine learning on multicore," in *Advances in Neural Information Processing Systems 19*, B. Schölkopf, J. Platt, and T. Hoffman, Eds. Cambridge, MA: MIT Press, 2007, pp. 281–288.

[13] C.-C. Chang and C.-J. Lin, "Libsvm: A library for support vector machines," *ACM Trans. Intell. Syst. Technol.*, vol. 2, no. 3, pp. 27:1–27:27, May 2011. [Online]. Available: http://doi.acm.org/10.1145/1961189.1961199

[14] A. Genkin, D. D. Lewis, and D. Madigan, "Large-scale bayesian logistic regression for text categorization," *Technometrics*, vol. 49, pp. 291–304(14), August 2007. [Online]. Available: http://www.ingentaconnect.com/content/asa/tech/2007/00000049/00000003/art00007

Yanan Xiao A first year master student as well as IEEE student member in CIS program, Masdar Institute. He loves programming when all the coursework is finished. When he feels tired of programming, he would read some books.

Aziza Al Sawafi First year Computing and Information Science student at Masdar Inst., got a bachelor degree in Network Engineering (United Arab Emirates University). Sport, drawing, designing, blogging, reading poems, photography, and riding horse/bicycle are my interests beside all things that are related to networking and computer science.