University of Delaware
Department of Electrical and Computer Engineering
ELEG652 - Principles of Parallel Architectures
Fall 2014
Prof. Guang R. Gao

Course Project:
A Symbiotic Relation between Big Data and Big Compute:
The Fraud Detection Case Study – Stage 3 Update

Description:

As some of you may have heard during one of this semester's Distinguished Lectures, DOE's Lucy Nowell mentioned that there exists a symbiotic relation between the emerging field of **Big Data** and the established field **Big Computing** (i.e. Supercomputing). In this project, you will learn such a close relationship through a specific case study: ***On-Line Fraud Detection***.

On-Line Fraud Detection involves identifying a fraud as soon as it has been perpetrated into the system. This technique is only successful by having a training algorithm that can produce a model suitable to be used by a real-time detector. In this project, we will focus on fraud detection for credit card transactions (see Figure 1), using ***Markov chains*** to train the model off-line and a ***parallel implementation*** of a concurrent queue for on-line detection.

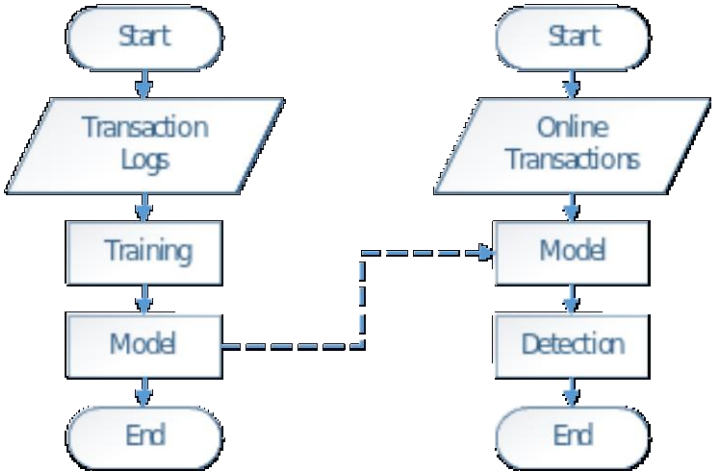

Fig 1. Offline training and online detection

The project will be divided in three stages to give the opportunity to students to make partial progresses towards a final presentation and report due at the end of the semester. This document presents the general project's overview and Stage 1's description. Other stage's descriptions will be posted in upcoming weeks.

Students will be divided in the following ***three groups***, with each group having ***two mentors*** and a ***leader*** (which will be nominated by the groups themselves). It is expected students have weekly meetings with their mentors to address any questions and doubts about the project.

The groups and their mentors are as follows:

Group 1: *Prof. Gao (ggao.capsl@gmail.com) and Rahul Deore (rahuld@udel.edu)*

- Weiyang Chen (wienchen@udel.edu)
- Yifeng Cong (cong@udel.edu)
- Zhihang Guo (zhguo@udel.edu)
- Jose Monsalve (josem@udel.edu)

Group 2: *Haitao Wei (whtaohust@gmail.com) and Sergio Pino (sergiop@udel.edu)*

- James Evangelos (jime@udel.edu)
- Minghao Li (marklee@udel.edu)
- Sun Ming (sunming@udel.edu)
- Siddhisanket Raskar (sraskar@udel.edu)

Group 3: *Stephane Zuckerman (szuckerm@udel.edu) and Tu Hao (tuhao@hust.edu.cn)*

- Laura Rozo (lrozo@udel.edu)
- Enbo Wang (enbowang@udel.edu)
- Chaunzhen Wu (kavfwu@udel.edu)
- Pouya Fotouhi (pfotouhi@udel.edu)

The stages for the project are the following:

Stage 1:

- *Title:* **Literature Review**

- *Objective:* Students will perform a background review on two relevant topics: Markov models and Hadoop framework. To be more efficient, each group may divide itself in sub teams of 2 students for this review. Although each sub team will take charge of one topic, the results of their reviews should be communicated internally to all group's members.

- *Result:* Students will submit the answers to the following questions before October 31st at 23.59.

  Markov Models:

  A Markov chain (MC) is a stochastic process with the Markov property. The term "Markov chain" refers to the sequence of random variables such a process moves through, with the Markov property defining serial dependence only between adjacent periods (as in a "chain"). It can thus be used for describing systems that follow a chain of linked events, where what happens next depends only on the current state of the system [1].

  This project trains a MC (unsupervised learning) to create a model that a fraud detection system can use to identify potential transaction outliers. The resulting model will be used to classify sequence of transactions as normal or anomalous.

In this part of the project your job is to get familiar with the basic concepts of Markov Chain and its application for this particular implementation of fraud detection. This process includes the reading of scientific papers, the proposal of a serial pseudo code for both the training and classification, the answering to some questions, and the creation of a report.

1. Background
    a. Read the following documents
        i. Reference [2] is the main reference for the implementation of both the training and the classification. In reading this reference, you may wish to focus on section 2 for the outline of the methodology, section 3.1 for the training of the model, and section 3.2 for the classifier.
        ii. Additional references for estimating the Markov transition matrix are provided in [3], [4]. In reference [3], focus on section 2.A (Transition Matrices When Individual Transitions Known) and appendix A (Estimating Transition Probabilities with Observed Transitions) for the training of the model. In reference [4], focus on section 1.2 (Markov chains) and 1.3 (Training algorithm).

    b. 1.2. For the report
        i. Write down a small summary about Markov chain. Within your summary take into account:
            1. Definition
            2. Properties
            3. Relationship between past, present, and future (states)
            4. Transition matrix
            5. Representation as a graph (vertex and edges)
        ii. With your own words explain the methodology presented in [2] for fraud detection.
        iii. With your own words explain the process to train the Markov chain.
        iv. With your own words explain the local classifier and the miss-rate metric.
        v. When possible add some working example to make clear your explanations.

2. From the readings, propose a serial pseudo code for the following:

    a. Training the model (estimating the Markov chain transition matrix).
    b. Creating a local classifier when the miss-rate metric is used.

    Explain clearly the steps of the algorithms and refer to the background concepts.

3. Answer the following question about Markov Chains:
    a. Decide whether each of the following vectors could be a probability vector.
        i. [0.4, 0.2, 0]
        ii. [¼, ⅛, ⅝]
        iii. [0.07, 0.04, 0.37, 0.52]

iv. [0.3, -0.1, 0.8]

b. Decide whether each of the following matrices could be a transition matrix, by definition. Sketch a transition diagram for any transition matrices.

$$\begin{bmatrix} 0.5 & 0 \\ 0 & 0.5 \end{bmatrix}$$

$$\begin{bmatrix} \frac{2}{3} & \frac{1}{3} \\ 1 & 0 \end{bmatrix}$$

$$\begin{bmatrix} \frac{1}{4} & \frac{3}{4} \\ \frac{1}{2} & \frac{1}{2} \end{bmatrix}$$

$$\begin{bmatrix} \frac{1}{4} & \frac{3}{4} & 0 \\ 2 & 0 & 1 \\ 1 & \frac{2}{3} & 3 \end{bmatrix}$$

c. Think about the following sentence: "*for a Markov chain, the conditional distribution of any future state Xn+1, given the past states X0 , X1 , . . . , Xn−1 and the present state Xn , is independent of the past states and depends only on the present state*." [5]. What are the implications of this sentence when designing the training algorithm? HINT: consider a toy training sequence of states $Q = q_1, q_2, \ldots, q_L$, and how the training algorithm should "analyze" the sequence.

References

[1] Markov chain. (2014, October 16). In Wikipedia, The Free Encyclopedia. http://en.wikipedia.org/w/index.php?title=Markov_chain&oldid=629904997

[2] S. Jha, K. Tan, and R. A. Maxion, "Markov chains, classifiers, and intrusion detection," in Proceedings. 14th IEEE Computer Security Foundations Workshop, 2001., 2001, pp. 206–219. http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=930147&tag=1

[3] Jones, Matthew T. Estimating Markov transition matrices using proportions data: an application to credit risk. International Monetary Fund, 2005.https://www.imf.org/external/pubs/cat/longres.aspx?sk=18387.0

[4] http://www.eecis.udel.edu/~lliao/cis841s06/hmmtutorialpart1.pdf

[5] Ross, Sheldon M. Introduction to probability models. Academic press, 2006.

Hadoop:

Apache Hadoop is an open-source software framework for storage and large-scale processing of data-sets on clusters of commodity hardware. MapReduce is a software framework for easily writing applications which process vast amounts of data in-parallel on large clusters (thousands of

nodes) in a reliable, fault-tolerant manner. Please follow installation tutorial [1] and set up Hadoop (version 2.5.0) environment on a single-node in a pseudo-distributed mode. Read the MapReduce paper [2] and tutorial [3] for basic understanding, and [4] for more details if necessary. Compile and run the WordCount v1.0 example in [3].

(1) Explain using your own words MapReduce programming model, using WordCount as example.

(2) Submit a WordCount v1.0 source code with your comments;

(3) - Considering the API definition for a Mapper and a Reducer, What are the data type requirements for a key and value Class?
- Should the Key and Value class implement a special interface?
- What data types can be used directly as a key or value, is it possible to use a list as a key or value?

(4) - What are combiners?
- When should I use a combiner in a MapReduce Job?
- If you remove the use of the combiner (job.setCombinerClass(IntSumReducer.class)) in the example source code, Is there any performance difference?
- Why does it work using the Reducer class as combiner in our example source code?

References:

[1]http://hadoop.apache.org/docs/current/hadoop-project-dist/hadoop-common/SingleCluster.html
[2]Dean, Jeffrey and Sanjay Ghemawat. "MapReduce: simplified data processing on large clusters." Communications of the ACM 51.1 (2008): 107-113. http://static.googleusercontent.com/media/research.google.com/en/us/archive/mapreduce-osdi04.pdf
[3]MapReduce Tutorial. http://hadoop.apache.org/docs/current/hadoop-mapreduce-client/hadoop-mapreduce-client-core/MapReduceTutorial.html
[4] Tom White."Hadoop: The Definitive Guide", 3rd Edition, Yahoo Press, 2012.

Stage 2:

- *Title:* **Design of a Parallel Version of the Markov Model based Learning Phase (See Fig. 1)**

Objective: Students will use the background knowledge and jargon acquired in Stage 1 to design a parallel version of the Markov model's training phase. This design assumes students will use the Mapreduce programming model, thus a clear mapping of the algorithm in terms of map and reduce functions is expected. The deliverables for this phase include:

1. A working serial Java/C code, with comments, of the training algorithm using the training sequence in the Markov Chains tutorial's slides #10 as the input. The result is supposed to be the same as the one presented in the tutorial, and the output should include:
    - The count for each state transition.
    - The transition matrix.

2. A pseudo code of the training algorithm, using the MapReduce programming model. Thus, a clear mapping of the algorithm in terms of map and reduce functions is expected. The pseudo code should provide:
   - The map and reduce functions, which implement the Markov chain training algorithm.
   - The input and output key-value data type for each map and reduce function.
   - The design should be accompanied with a clear definition of the design, diagrams, and references.

3. Extra credit will be given if students make:
   a. The pseudo code work with any given input file. A working Java code with comments is expected to submit.
   b. A dataflow representation of the Mapreduce pseudo code. In this view, each node represents either a map or a reduce function, and the edges represent how the data flow across the nodes. One possible example can be found in the Tez apache project at http://tez.apache.org/index.html

HINTS:

- During the training you are provided with the set of states. For example, in the tutorial we knew that the states were (R)aining and (S)unny.
- The students will be provided with the input dataset for Mapreduce. Each file has one sequence per line, and each sequence is assumed to be generated by the same transition matrix.
- ***MapReduce Installation:*** It is easier to install Hadoop on a CentOS virtual machine. Students can follow the MapReduce tutorial slides page #23 to set up an all-in-one VM with Hadoop and Eclipse already installed. These are the steps:

    - Download and install VirtualBox on your machine: http://virtualbox.org/wiki/Downloads
    - Download the Cloudera Quickstart VM for virtualbox at http://www.cloudera.com/content/cloudera/en/downloads.html
    - Uncompress the VM archive. It is compressed with 7-Zip. If needed, you can download a tool to uncompress the archive at http://www.7-zip.org/
    - Start VirtualBox(sudo virtualbox) and click Import Appliance. Click the folder icon beside the location field. Browse to the uncompressed archive folder, select the .ovf file, and click the Open button. Click the Continue button. Click the Import button.

References:

Please refer to the slides for useful references.

- *Result due on Due on November 7th at 23.59.*

Stage 3:

- *Title:* **Implement your design from Stage 2 into a Fraud-Detection Platform 652-FDP**

- *Objective:* 652-FDP is a MapReduce implementation which trains a general model in order to produce a transition matrix that describes all the customers (just one 18x18 matrix). After you get familiar with 652-FDP, you will need to integrate your own design, and in addition, expand it in order to train a Markov model that represents the profile of each customer (one transition matrix per customer).

[Download ELEG652-stage3-652FDP.zip from the course webpage] 652-FDP is a MapReduce implementation of the basic Markov Model training phase in fraud detection. Along with the 652-FDP, a dataset is provided for training.

[Download ELEG652-stage3-dataset.zip from the course webpage] The dataset is a set of transaction log files, the format of the log record is:

Customer_id, Transaction_id, Transaction_type

The transaction id values are unique and have the semantics of positioning a transaction along a time line. Each log file will contain some transactions of different customers. For example:

TJS41P2U9A, 97351659821043, LHS
R8NYNNV4NP, 97358894512859, MHN
WRBQZUT3G9, 97359505201847, LNS
NLJT50JA7K, 97363549658528, HHN
0VT2QJN9Q2, 97370708307103, LHN

Each transaction is encoded by the following three quantities and expressed as a 3-letter token. These three quantities together are referred as the transaction type. Thus, there are 18 (3 x 2 x 3) possible transaction types (we consider the transaction types as the states in the Markov model).

Amount spent: Low, Normal, or High
Whether the transaction includes high price ticket item: Normal or High
Time elapsed since the last transaction: Large, Normal, or Small

[Project] 652-FDP will generate an 18 x 18 state transition probability matrix from the training data.

As Figure 2 shows, there are 9 source files in the project. There are 2 MapReduce jobs, which are SequenceBuilderMap/Red and MarkovChainModelMap/Red. The first job "SequenceBuilderMap/Red" builds per customer the sequence of transactions (sorted by time) from the log files. The second job counts the sampled transitions and estimate the transition matrix (call TransitionMatrix.java to compute the transition matrix). CompositeKey and TransitionWritable are used to serialize data. MCTraning is the main program and is in charge of sending the two jobs to the hadoop cluster.

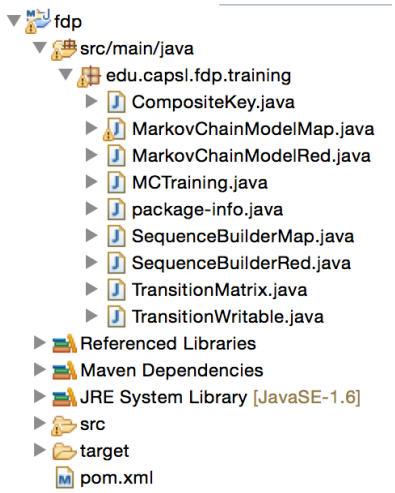

Fig 2. 652-FDP project in Eclipse

The code that should be of most interest to you is (HINTS):

- ***SequeceBuilderMap/SequenceBuilderRed:*** In this phase you have all of the information about the transactions. Ask yourself how the code is grouping the data after the mapper and what the semantics of the iterable that represents the values at the reducer are.
- ***MarkovChainModelMap/MarkovChainModelRed:*** In this phase it is assumed that the input set is composed by sampled transitions (no customer or time is present on the records), and the code is just estimating one transition matrix. Ask yourself how you can add the customer information to this process and how your design can increase the parallelism in the reduce phase (in the case when you have to estimate one transition matrix per customer).
- ***MCTraining:*** Main class that schedule the execution. Ask yourself how you can improve the execution time of the internal jobs, think in your first report and in the questions related to the performance of word count. How many reducers your jobs are using? How can you tell Hadoop to increase the parallelism in the reduce phase?
- ***TransitionMatrix:*** data structure to support the estimation.

[Compile/Run/IDE]
1. **Compile**: go to the 652-FDP project folder in your terminal and type "mvn package". It is assumed that you have maven. This will create a folder called target and the file you want to take care of is "target/fdp-0.1-job.jar"

2. **Run with hadoop**: Hadoop jar target/fdp-0.1-job.jar /path/to/input/dataset /path/to/output/folder. The paths are supposed to be in HDFS.

3. **If you want to use eclipse for development**: then create a project with "mvn eclipse:eclipse" and then import the project. Also, to fix the classpath is "mvn eclipse:configure-workspace"

Acceptance Test:

1. **An 18 x 18 transition matrix, which is the 652-FDP output for the given dataset.** Read the source code, understand how the Markov Chain training works in MapReduce mode, and especially pay attention to the hints. Learn how to compile the source code using both maven and eclipse. After you get the

executable jar file, run it with Hadoop using given dataset, submit the result transition matrix (one 18 x 18 matrix).

2. **N 18 x 18 transition matrixes for N customers, which is your project output for given dataset.** Replace the training session of the code by your own design on stage 2, and in addition expand your design in order to train a Markov model that represents the profile of each customer. Note that you should revise, and refine your pseudo code for this purpose. All the source codes should be with comments. The output format should be one transition matrix for each customer_id (you can encode the transition matrix as a 1d array):

Customer_id1: Transition Matrix1
Customer_id2: Transition Matrix2
……

The output file will be later used for online detection. Thus, an online detection system will use this output file to detect the outlier.

3. **A project with all the necessary source code and configuration files.** Each group will have an account on our cluster. Please write, compile and debug in your own development environment, and then upload your project to your home directory, name it "652-FDP-Group-X". The source codes in the project are supposed to be complied using "mvn package". The executable jar file will be executed with Hadoop cluster with given dataset, and the output transition matrices are supposed to be almost the same with the matrices, which are used to generate the dataset.

Each group will have the CAPSL cluster available in the following slots:

- Friday Nov 14th from 00.00 to 23.59: ***Group 1.***
- Saturday Nov 15th from 00.00 to 23.59: ***Group 2***.
- Sunday Nov 16th from 00.00 to 23.59: ***Group 3.***
- Monday Nov 17th from 00.00 to 23.59: ***Group 1.***
- Tuesday Nov 18th from 00.00 to 23.59: ***Group 2.***
- Wednesday Nov 19th from 00.00 to 23.59: ***Group 3.***
- Thursday Nov 20th from 06.00 to 11.59: ***Group 1.***
- Thursday Nov 20th from 12.00 to 17.59: ***Group 2.***
- Thursday Nov 20th from 18.00 to 23.59: ***Group 3.***

An email to each group leader will be sent with the instructions to access the machine.

- *Result due on Due on November 21st*.

Stage 4:

- *Title:* **Project's Presentation and Report**

- *Objective:* Present each group's results to the class.

- *Result due on Due on December 2nd and December 5th:* TBA.

Submission Details:

All reports will be submitted using an IEEE Paper Template (http://www.ieee.org/conferences_events/conferences/publishing/templates.html). Remember to cite all your sources and to include all the group's member's names in the reports and presentations.

Include any source files you wrote. Each program you write must be commented and have its own Makefile or equivalent.

Remember to include the HW/SW specifications of the machine(s) where you run your experiments.

All submissions will be made by the group's leader to Jaime Arteaga jaime@udel.edu as following: all files in a ZIP file with the name ***group<YOUR_GROUPS_NUMBER>-project-stage<NUMBER_OF_STAGE>-eleg652-14f.zip*** (e.g. *group2-project-stage3-eleg652-14f.zip*) with subject ***ELEG652-14F Project Stage <NUMBER_OF_STAGE>*** (e.g. *ELEG652-14F Project Stage 3*) before the specified deadline.