CSE6250: Big Data Analytics in Healthcare
Homework 3

Jimeng Sun

Deadline: Oct 8, 2017, 11:55 PM AoE

- Discussion is encouraged, but each student must write his/her own answers and explicitly mention any collaborators.
- Each student is expected to respect and follow the GT Honor Code.
- Please type the submission with LaTeX or Microsoft Word. We **will not** accept hand written submissions.
- Please **do not** change the filenames and function definitions in the skeleton code provided, as this will cause the test scripts to fail and subsequently no points will be awarded.

Overview

Accurate knowledge of a patient's disease state is crucial which requires knowing accurate phenotypes about patients based on their electronic health records. There are several strategies for phenotyping, including supervised rule-based methods as well as unsupervised methods. In this homework, you will implement both type of phenotyping algorithms. You will be required to implement these using Spark.

Prerequisites [0 points]

This homework is primarily about using Spark with Scala. You can download and install Spark on your local machine by following instructions at http://spark.apache.org/docs/1.3.1/ (tested on 1.3.1 $\sim$ 1.6.2, and 2.0 has not compatible API with our code) if you want to to try spark-shell. Alternatively, you can use one of our virtual environments you used for the lab sessions from http://www.sunlab.org/teaching/cse8803/fall2016/lab/environment/. Programming assignments in this homework actually do not require Spark installation since we use SBT build tool http://www.scala-sbt.org/. You can also

develop code by using an IDE like IntelliJ (https://www.jetbrains.com/idea/) for your convenience (free community edition is enough for developing homework code, but as a student, you can get education license for full edition) or by using a Notebook platform such as Jupyter or Zeppelin. We provide a virtual environment with pre-installed Jupyter and Zeppelin (http://www.sunlab.org/teaching/cse8803/fall2016/lab/env-docker/) for those interested in.

For programming problem, you will be given the code skeleton and test cases in this zip file.

Then you need to download data from S3, unzip that and put that into your code directory

```
cd code
wget https://s3.amazonaws.com/cse8803bdh/hw3/data.tar.gz
tar -zxvf data.tar.gz
```

Note that the data folder should be inside your code folder.

If you are a mac user you should be able to use below command to compile and run the code by

```
sbt/sbt compile run
```

And to run the test cases:

```
sbt/sbt compile test
```

Otherwise, you will need to refer to SBT installation manual to update *sbt/sbt* script first. Then you can call in above way.

1 Programming: Rule based phenotyping [30 points]

Phenotyping can be done using a rule-based method. The Phenotype Knowledge Base (PheKB) (https://phekb.org) provides a set of rule-based methods (typically in the form of decision trees) for determining whether or not a patient fits a particular phenotype.

In this assignment, you will implement a phenotyping algorithm for type-2 diabetes based on the flowcharts below. The algorithm should

- Take as input event data for diagnoses, medications, and lab results.
- Return an RDD of patients with labels (*label*=1 if the patient is case, *label*=2 if the patient is control, *label*=3 otherwise).

You will implement the *Diabetes Mellitus Type 2* algorithms from PheKB. We have reduced the rules for simplicity. Thus, you should follow the simplified flowchart provided for your homework, but can refer to http://jamia.oxfordjournals.org/content/19/2/219.long for more details if desired.

The following files in *data* folder will be used as inputs:

- **encounter_INPUT.csv**: Each line represents an envounter. The encounter ID and the patient ID (Member ID) are separate columns. *Hint: sql join*
- **encounter_dx_INPUT.csv**: Each line represents an encounter. The disgnoses (ICD9 codes) are in this file.
- **medication_orders_INPUT.csv**: Each line represents a medication order. The name of medication is found in one of the columns on this file.
- **lab_results_INPUT.csv**: Each line represents a lab result. The name of the lab (use 'Result_Name' column), the units for the lab, and the value for the lab are found in specific columns on this file.

For your project, you will load input CSV files from `<your_code_project_root>/data/<some_input.csv>`. You are responsible for transforming the above CSV files into RDDs.

The simplified rules which you should follow for phenotyping of Diabetes Mellitus Type 2 are shown below. These rules are based off of the criteria from the PheKB phenotypes, which have been placed in the folder `/phenotyping_resources/`.

- **Requirements for Case patients**: Figure 1 details the rules for determining whether a patient is case. Certain parts of the flowchart involve criteria that you will find in the handout folder `/phekb_criteria/`.
  - `/phekb_criteria/T1DM_DX.csv`: Any of the ICD codes present in this file will be sufficient to result in `YES` for the Type 1 DM diagnosis criteria.
  - `/phekb_criteria/T1DM_MED.csv`: Any of the medications present in this file will be sufficient to result in `YES` for the Order for Type 1 DM medication criteria. Please also use for the criteria Type 2 DM medication precedes Type 1 DM medications.
  - `/phekb_criteria/T2DM_DX.csv`: Any of the ICD codes present in this file will be sufficient to result in `YES` for the Type 2 DM diagnosis criteria.
  - `/phekb_criteria/T2DM_MED.csv`: Any of the medications present in this file will be sufficient to result in `YES` for the Order for Type 2 DM medication criteria. Please also use for the criteria Type 2 DM medication precedes Type 1 DM medications.

- **Requirements for Control patients**: Figure 2 details the rules for determining whether a patient is control. Certain parts of the flowchart involve criteria that you will find the handout folder `/phekb_criteria/`.
  - `/phekb_criteria/ABNORMAL_LAB_VALUES_DX.csv`: You can refer to these abnormal lab values criteria for controls.

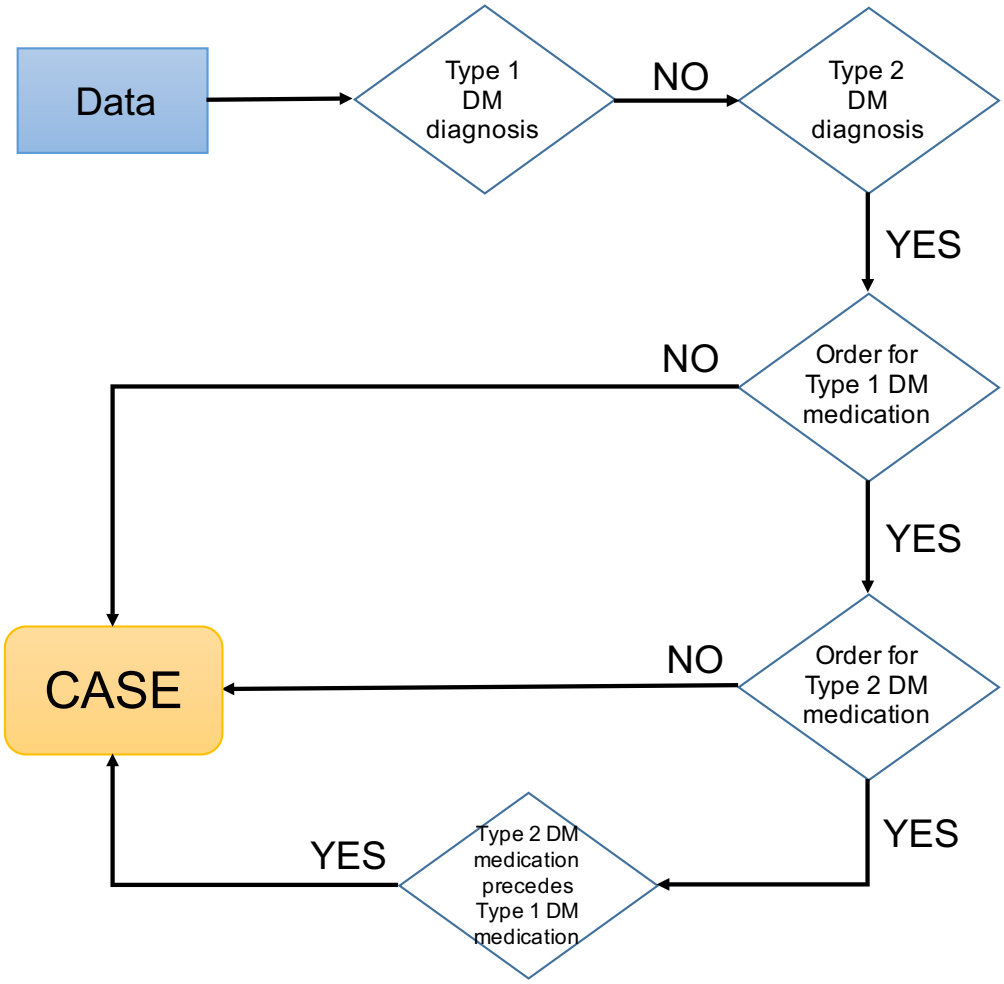

Figure 1: Determination of cases

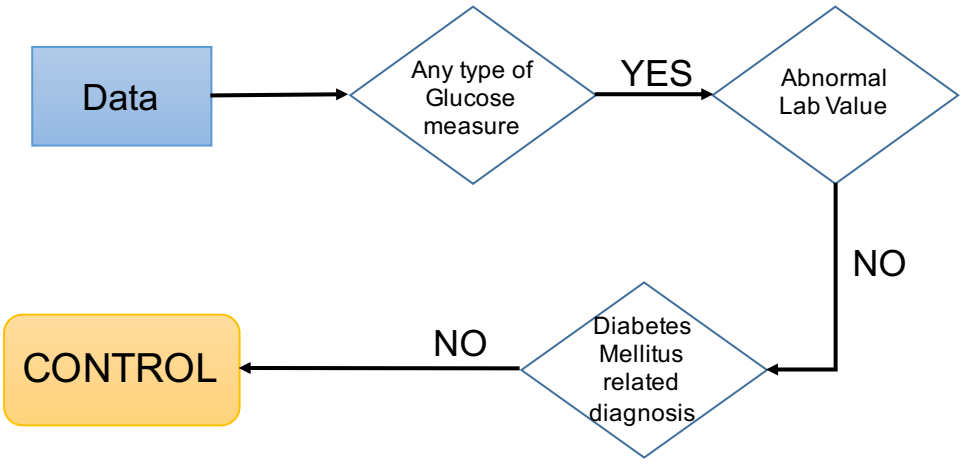

Figure 2: Determination of controls

– `/phekb_criteria/DM_RELATED_DX.csv`: Any of the ICD codes present in this file will be sufficient to result in `YES` for the Diabetes Mellitus related diagnosis cri-

teria.

In order to help you verify your steps, expected counts along the different steps have been provided in:

- `/phenotyping_resources/expected_count_case.png`
- `/phenotyping_resources/expected_count_control.png`

Additional hints and notes are at times provided directly in the code comments at the locations of relevance, please read these carefully when provided.

a. Implement *edu.gatech.cse8803.main.Main.loadRddRawData* to load INPUT CSV files in data folder as structured RDD. Follow instructions for turning this in.[5 points]

b. Implement *edu.gatech.cse8803.phenotyping.T2dmPhenotype*. Follow instructions for turning this in.

- Finding case patients [10 points]
- Finding control patients [10 points]
- Finding other patients [5 points]

2 Programming: Unsupervised phenotyping via clustering [45 points]

At this point you have implemented a supervised, rule-based phenotyping algorithm. Those type of methods are great for picking out specific diseases, in our case diabetes and rheumatoid arthritis. However, they are not good for discovering new, complex phenotypes. Such phenotypes can be disease subtypes (i.e. severe hypertension, moderate hypertension, mild hypertension) or they can reflect combinations of diseases that patients may present with (e.g. a patient with hypertension and renal failure).

2.1 Feature Construction [16 points]

Given the raw data, you need to start with feature construction. You will need to implement ETL using Spark with similar function as what you did in last homework using Pig. Given that you know which diagnoses (in the form of ICD-9 codes) each patient exhibits, and which medication each patient took, these can be used as features in a clustering model. Using the RDDs that you created in *edu.gatech.cse8803.main.Main.loadRddRawData*, you need to construct features using COUNT aggregation for medication and diagnostics, AVERAGE aggregation for lab test values.

a. Implement feature construction in

edu.gatech.cse8803.features.FeatureConstruction. Implement two kinds of feature construction, one constructs features using all available icd codes, lab and medication, and

another with only features related to the phenotype. See comments of the source code for details.

2.2 Evaluation Metric [8 points]

Purity is a metrics to measure the quality of clustering, it's defined as

$$purity(\Omega, C) = \frac{1}{N} \sum_{k} \max_{j} |w_k \cap c_j|$$

where N is the number of samples, k is index of clusters and j is index of class. w_k denotes the set of samples in k-th cluster and c_j denotes set of samples of class j.

a . Implement *edu.gatech.cse8803.clustering.Metrics*.

In this homework you will perform some clustering using Spark. Spark contains MLLib library with implementation of the k- means clustering algorithm and the Gaussian Mixture Model algorithm.

From clustering, we can discover groups of patients with similar characteristics. Please cluster the patients based upon diagnoses, labs and medications. If there are d distinct diagnoses, l distinct medications and m medications, then there should be $d + l + m$ distinct features.

2.3 K-Means Clustering [5 points]

a. Implement k-means clustering for $k = 3$. Follow the hints provided in the skeleton code in *edu.gatech.cse8803.main.Main.scala:testClustering*.

b. Compare clustering for the $k = 3$ case with the ground truth phenotypes that you computed for the rule-based PheKB algorithms. Specifically, for each of *case*, *control* and *unknown*, report the percentage distribution in the three clusters for the two feature construction strategies. Report the numbers in the format shown in Table 1 and Table 2.

Percentage Cluster	Case	Control	Unknown
Cluster 1	x%	y%	z%
Cluster 2	xx%	yy%	zz%
Cluster 3	xxx%	yyy%	zzz%
	100%	**100%**	**100%**

Table 1: Clustering with 3 centers using all features

2.4 Clustering with Gaussian Mixture Model (GMM) [5 points]

a. Implement GaussianMixture for $k = 3$. Follow the hints provided in the skeleton code in *edu.gatech.cse8803.main.Main.scala:testClustering*.

Percentage Cluster	Case	Control	Unknown
Cluster 1	x%	y%	z%
Cluster 2	xx%	yy%	zz%
Cluster 3	xxx%	yyy%	zzz%
	100%	**100%**	**100%**

Table 2: Clustering with 3 centers using filtered features

b. Compare clustering for the $k = 3$ case with the ground truth phenotypes that you computed for the rule-based PheKB algorithms. Specifically, for each of *case*, *control* and *unknown*, report the percentage distribution in the three clusters for the two feature construction strategies. Report the numbers in the format shown in Table 1 and Table 2.

2.5 Clustering with Streaming K-Means [5 points]

a. When data arrive in a stream, we may want to estimate clusters dynamically, updating them as new data arrive. *Spark.mllib* provides support for streaming k-means clustering. The algorithm uses a generalization of the mini-batch k-means algorithm incorporating **forgetfulness**. Show why we can use streaming K-Means by deriving its update rule and describe how it works regarding pros and cons, then discuss the impact of forgetfulness value on balancing the relative importance of new data arrive versus past history.
b. Implement StreamingKMeans algorithm for $k = 3$. Follow the hints provided in the skeleton code in *edu.gatech.cse8803.main.Main.scala:testClustering*.
c. Compare clustering for the $k = 3$ case with the ground truth phenotypes that you computed for the rule-based PheKB algorithms. Specifically, for each of *case*, *control* and *unknown*, report the percentage distribution in the three clusters for the two feature construction strategies. Report the numbers in the format shown in Table 1 and Table 2.

2.6 Discussion on k-means and GMM [6 points]

a. Briefly discuss what you observe in 2.3b and 2.4b.
b. Re-run k-means and GMM from the previous two sections for different k (you may run it each time with different k). Report purity for filtered and all features for each k by filling up Table 3. Discuss patterns observed, if any.
Change back k to 3 in your final code deliverable.

k	K-Means All features	K-Means Filtered features	GMM All Features	GMM Filtered features
2				
5				
10				
15				

Table 3: Purity values for different number of clusters

3 Advanced phenotyping with NMF [20 points]

Given a feature matrix $\boldsymbol{V}$, the objective of NMF is to minimize the Euclidean distance between the original non-negative matrix $\boldsymbol{V}$ and its non-negative decomposition $\boldsymbol{W} \times \boldsymbol{H}$ which can be formulated as

$$\underset{\boldsymbol{W} \geq 0, \boldsymbol{H} \geq 0}{\operatorname{argmin}} \quad \frac{1}{2}||\boldsymbol{V} - \boldsymbol{W}\boldsymbol{H}||_2^2 \tag{1}$$

where $\boldsymbol{V} \in \mathbb{R}_{\geq 0}^{n \times m}$, $\boldsymbol{W} \in \mathbb{R}_{\geq 0}^{n \times r}$ and $\boldsymbol{H} \in \mathbb{R}_{\geq 0}^{r \times m}$. $\boldsymbol{V}$ can be considered as a dataset comprised of n number of m-dimensional data vectors, and r is generally smaller than n.

To obtain a $\boldsymbol{W}$ and $\boldsymbol{H}$ which will minimize the Euclidean distance between the original non-negative matrix $\boldsymbol{B}$, we use the Multiplicative Update (MU). It defines the update rule for $\boldsymbol{W}_{ij}$ and $\boldsymbol{H}_{ij}$ as

$$\boldsymbol{W}_{ij}^{t+1} = \boldsymbol{W}_{ij}^{t} \frac{(\boldsymbol{V}\boldsymbol{H}^\top)_{ij}}{(\boldsymbol{W}^t\boldsymbol{H}\boldsymbol{H}^\top)_{ij}}$$
$$\boldsymbol{H}_{ij}^{t+1} = \boldsymbol{H}_{ij}^{t} \frac{(\boldsymbol{W}^\top\boldsymbol{V})_{ij}}{(\boldsymbol{W}^\top\boldsymbol{W}\boldsymbol{H}^t)_{ij}}$$

Pseudo-code for the rule is listed below.

```
Initialize W, H randomly;
repeat
    /* Updating W[i,:]                                          */
    Save HH^T as a global variable H_s;
    W^{t+1}[i,:] = W^t[i,:] ⊙ V[i,:]H^T ⊙ (W^t[i,:]H_s)^{-1}
    /* Updating H[:,i]                                          */
    Save W^T W as a global variable W_s;
    H^{t+1}[:,i] = H^t[:,i] ⊙ W^T V[:,i] ⊙ (W_s H^t[:,i])^{-1}
until (1/2)||V - WH||_2^2 < ε;
```

You will decompose your feature matrix $\boldsymbol{V}$, from **2.1**, into $\boldsymbol{W}$ and $\boldsymbol{H}$. In this equation, each row of $\boldsymbol{V}$ represents one patient's features and a corresponding row in $\boldsymbol{W}$ is the patient's

cluster assignment, similar to a Gaussian mixture. For example, let $r = 3$ to find three phenotype(cluster), if row 1 of $\boldsymbol{W}$ is $(0.23, 0.45, 0.12)$, you can say this patient should be group to second phenotype as 0.45 is the largest element.

$\boldsymbol{W}$ can be very large, i.e. billion patients, which must be worked on in a distributed fashion while $\boldsymbol{H}$ is relatively small and can fit into a single machine's memory. You will these two types of matricies as distributed RowMatrix and local dense Matrix respectively in the skeleton code.

a. Implement the algorithm, as previously described, in *edu.gatech.cse8803.clustering.NMF*. [15 points]

b. Run NMF clustering for $k = 2, 3, 4, 5$ and report the purity for two kinds of feature construction. [5 points]

c. Perform the comparison of clustering for the $k = 3$ case with the ground truth phenotypes that you computed for the rule-based PheKB algorithms. Specifically, for each cluster, report the percetage of *case*, *control* and *unknown* in Table 4 and Table 5 for two feature construction strategies. [5 points]

d. Show why we can use MU update rule by deriving the equation for it. [10 points bonus]

Percentage Cluster	Case	Control	Unknown
Cluster 1	x%	y%	z%
Cluster 2	xx%	yy%	zz%
Cluster 3	xxx%	yyy%	zzz%
	100%	**100%**	**100%**

Table 4: NMF with 3 centers characteristics using all features

Percentage Cluster	Case	Control	Unknown
Cluster 1	x%	y%	z%
Cluster 2	xx%	yy%	zz%
Cluster 3	xxx%	yyy%	zzz%
	100%	**100%**	**100%**

Table 5: NMF with 3 centers characteristics using filtered features

4 Submission [5 points]

The folder structure of your submission should be as below or your code will not be graded. You can display fold structure using *tree* command. All other unrelated files will be discarded during testing. You may add additional methods, additional dependencies, but make sure existing methods signature doesn't change. It's your duty to make sure your code is compilable with provided sbt. Be aware that writeup is within code root.

```
<your gtid>-<your gt account>-hw3
|-- homework3answer.pdf
|-- build.sbt
|-- project
|   |-- build.properties
|   \-- plugins.sbt
|-- sbt
|   \-- sbt
\-- src
    \-- main
        |-- java
        |-- resources
        \-- scala
            \-- edu
                \-- gatech
                    \-- cse8803
                        |-- clustering
                        |   |-- NMF.scala
                        |   |-- Metrics.scala
                        |   \-- package.scala
                        |-- features
                        |   \-- FeatureConstruction.scala
                        |-- ioutils
                        |   \-- CSVUtils.scala
                        |-- main
                        |   \-- Main.scala
                        |-- model
                        |   \-- models.scala
                        \-- phenotyping
                            \-- PheKBPhenotype.scala
```

Create a tar archive of the folder above with the following command and submit the tar file.

```
tar -czvf <your gtid>-<your gt account>-hw3.tar.gz \
  <your gtid>-<your gt account>-hw3
```