COMP-4960 Final Report

Dr. Alioune Ngom

Joel Rorseth

104407927

1.0 Introduction

Throughout the Fall 2018 and Winter 2019 semesters, I have been researching under Dr. Alioune Ngom of the University of Windsor Computer Science Department. Given his research interest in the field of Bioinformatics, I spent both semesters investigating and developing techniques to perform drug repositioning (also known as drug repurposing).

1.1 Problem Definition

To establish the goals of my research, we first define the problem at hand. Drug repositioning is the processes of re-investigating existing drugs (whether approved by standards organizations or not), and identifying candidate drugs that would be suitable to treat a different disease. In the context of my research, the focus will be on identifying drugs to treat breast cancer. Thus, the goal of my research is to identify these repurposed drugs, by utilizing machine learning techniques on genetic expression data. Correlation must be determined between the expression profiles of breast cancer disease samples, and these candidate drugs.

2.0 Input and Datasets

The following datasets are necessary input to the scripts written for this project.

2.1 Gene Expression Data

/Datasets/brca_metabric/data_RNA_Seq_expression_median.txt

This 24,374 x 1906 matrix *G* records the genetic expression of 24,374 genes, for 1904 different breast cancer (patient) samples. In *G*, the first two columns are the the identifier of the measured gene. Any *G*[*i,j*] stores the gene *i*'s expression signature value within patient *i*'s sample.

2.2 CNA Matrix

/Datasets/brca_metabric/data_CNA.txt

This 22,544 x 2175 matrix *C* records the discrete copy number data of 22,544 genes, for 2173 patients. The first two columns of *C* denote the identifier of the gene at that corresponding row. Any *C*[*i,j*] is the copy number of gene *i* in patient *j*'s sample, and indicates that gene *i* is *diploid* if the value is 0.

2.3 Breast Cancer Biomarker Network (By Subtype)

/Datasets/Ten-Network Biomarkers/Subtype-1.xls

For each subtype, there exists a .XLS representation of a graph (network) of genes. Each file is an *n* x 3 matrix *N* representing a weighted graph, where each row *i* is an edge, (*N*[*i*,1], *N*[*i*,2]) with weight *N*[*i*,3].

2.4 LINCS Breast Cancer Drug-Gene Z-scores

/Datasets/Lincs-Breast-Cancer.RData

The LINCS dataset is a 12,328 x 21,567 matrix *L,* representing the z-scores (of signatures) of 12,328 genes treated with 1710 different drugs. The 1710 unique drugs appear in many columns in the original LINCS file, each with varying perturbation dose, time, and cell id. A common

dose, time and cell id are established in the script (discussed later) to reduce to 1710 unique drug columns. Any given $L[i,j]$ stores a z-score for gene i when treated with drug j.

2.5 Drugbank.ca XML Database

/Datasets/full_drugbank.xml

Taken from Drugbank.ca, this open source database file is an XML formatted export of the website's current drug databank. Deeply nested within this file are the approval status(es) of all drugs listed within their database.

3.0 Obtaining Genetic Z-scores For Breast Cancer Subtypes

Given that we will be using machine learning techniques to suggest drugs for potential repurposing, we must consider the dimensionality of the data. Thus for each subtype, we employ several methods of removing gene measurements that are unimportant to this subtype. */Preprocessing/gene_filtering.r* . For each subtype, we do the following two steps.

3.1 Gene Filtering

To begin this filtering process, we shift our focus to the Gene Expression Data, in matrix G. Containing signatures of 24,374 genes for 1904 patient samples, G has many unimportant genes. To determine importance, we import the Breast Cancer Biomarker Network N (for the current subtype), and the CNA Matrix C. Since G contains the most genes, we reduce G by removing all rows (genes) which do not appear in C. This is necessary due to the fact that each gene will need a CNA record in order to determine its z-score in the next step. Next, to further refine the

biomarker network *N*, we remove all edges in N whose weight is less than 50%. *G* then removes all gene rows which do not appear in *N*. We have now reduced *G* to include only genes significantly important to this subtype. Depending on the breast cancer subtype, the gene expression data *G* is now reduced to 50-200 genes, from 24,374.

3.2 Calculating Z-scores

After filtering the genes in the gene expression matrix *G*, we are prepared to calculate the z-scores of the few hundred genes for each subtype. We aim to establish a new matrix *Z*, of identical dimensionality to that of *G*, where $Z[i,j] = z\text{-}score(G[i,j])$. In the script, we calculate z-scores row by row, for each gene in *G*.

For the current gene (row *i*) in *G*, we start by determining the patients within *C* for whom this gene was diploid. Specifically, a patient (represented within the *jth* column of *C*) is diploid for this gene if $C[i,j] = 0$. We then extract the expression data for gene *i* in *G*, only for this subset of diploid patients columns. This subset contains the patient samples that are useful in the z-score formula. As such, the script calculates the mean **μ** and standard deviation **σ** on the subset, for this gene. Following this, z-scores are calculated for each (unfiltered) *jth* patient $G[i,j]$ for the current *ith* gene, using the common formula $Z[i,j] = (G[i,j] - \mu) / \sigma$.

Having calculated a z-score for each patient's gene expression, we have essentially determined a z-score based representation of the genes important to each subtype. In order to compare these 10 diseases with the z-scores of drugs we will be investigating, we require a single vector for each

breast cancer subtype, listing a single z-score for each gene. Thus, we compute the average of the z-scores in *Z* for each corresponding gene, inserting them into a table mapped to their corresponding gene name. This *n* x 2 table is written to *subtype_s_zscores.csv* , for each subtype *s* for which *n* genes remained in *G*.

4.0 Creating the Z-score Matrix

To consolidate the breast cancer data (for each stage) and the drug data, the second script combines both into a single matrix *M*. Given that we computed z-scores for genes relevant to each subtype, we merge this data with drug z-scores taken from the LINCS Breast Cancer Drug-Gene Z-scores matrix *L*.

4.1 Removing Duplicate Drugs from LINCS

As mentioned in Section 2.4, the original LINCS z-score matrix is 12,328 (genes) x 21,567 (drug records). Upon further investigation, Only 1710 unique drugs are measured in the 21,567 columns, but were measured with varying parameters (perturbation dose, perturbation time, and perturbation cell id). Taking the most common experiment parameters, we filter to columns with a dose of *10.0 um*, time of *24 hr*, and cell id *MCF7*. This leaves approximately 100 columns sharing drug name and parameter values, which are then arbitrarily pruned by the script.

4.2 Combining Disease and Drug Z-scores

As input to this secondary script, we now have a reduced *L* of size 12,328 x 1710 (genes by drugs), and a vector *Di* of variable length *n* for each subtype *i*. For each breast cancer subtype *i*,

we load *Di* from file, and create reduced copies of both *L* and *Di* , containing only genes which are common to both (their intersection). Thus *Di* has new length *n'*, with *L* reducing to *n'* x 1710. At this point, $|Di| = n'$, allowing it to be appended to *L* as an additional column. The script performs this operation by inserting into the first column position (and shifting the columns) of *L*. The transpose of this merged matrix becomes *M*, which contains the z-scores for an important subset of genes for a given breast cancer subtype and 1710 drugs. *M* is written to *all_zscores_subtype_s.Rds*, for each subtype *s*.

5.0 Reducing the Set of Drugs

Entering the third script, each subtype has a matrix *M* with scores for the subtype disease and drugs from LINCS. In order to use machine learning to identify candidate drugs for repositioning, we must reduce each *M* further (to reduce dimensionality).

5.1 Building Anti-Correlation Matrix to Help Filter

To begin the final filtering process (this time for the drugs), the third script builds an anti-correlation matrix *A* for each *M*, by comparing drug z-scores (stored in $M[2\ldots n, j]$) for each gene with corresponding z-scores for the disease (stored in $M[1, j]$). Thus, given *M* of size *n* x *m* , we create *A* of size *(n-1)* x *m* by using the following assignment in a for loop ($\forall i > 2$):

$$A[i,j] = \begin{cases} +1 & \text{if M[i,j] > 0 and M[1,j] < 0} \\ -1 & \text{if M[i,j] < 0 and M[1,j] > 0} \\ 0 & \text{otherwise} \end{cases}$$

In the above piecewise assignment, the anti-correlation matrix $A[i,j]$ is assigned +1 if gene j indicates up-regulation when drug i is applied, or -1 if down-regulation occurs. These two instances are important, and are indicators that a given drug has a meaningful effect treatment for this breast cancer disease (subtype). With the remaining values being assigned 0, we similarly consider these gene / drug pairs to be useless. The script then proceeds to remove 50% of the worst rows (drugs) in *M,* where the worst are the (corresponding) rows which contain the most 0's in *A*. A similar filtering process is applied column-wise, where *k* of the worst columns are pruned (k being the number of 0's in the row with least number of 0's). Using this process, each *M* has been reduced in size to 855 x *m*, for some *m* genes that have survived these filtering steps. For each subtype *s, M* is written to *reduced_zscores_subtype_s.Rds*.

6.0 Extracting Drug FDA Approval Status

Before determining which drugs closely align with the breast cancer subtypes, we hope to augment the results by determining the FDA approval status of our LINCS drugs. Two Python scripts have been written to explore the Drugbank.ca XML database, and extract the approval status of drugs listed within. *extract_drugbank_approval_status.py* parses through the XML, writing a JSON mapping of Drugbank.ca drugs to their listed approval statuses ("approved", "illicit", "experimental", "withdrawn", "nutraceutical","investigational", or "vet_approved") into *drugbank_approval_status.json*. The second script *get_lincs_approval_status.py* cross references the Drugbank mapping with the 1710 LINCS drugs, writing a new mapping of LINCS drugs found in Drugbank, to their approval statuses, into *lincs_approval_status.json*. Of the 1710 LINCS drugs, 1086 of them are recorded in Drugbank and give us their approval status.

7.0 Clustering to Determine Drugs Close to Disease

In the final step of the project, for each breast cancer disease subtype, we consider plotting the disease and (n-1) drug rows from M (size n x m) in m dimensional space. The drugs closer to the disease point are proportionally more suitable for repurposing (for this subtype). To gauge proximity of the points, we may employ machine learning techniques to cluster and determine drugs with similar profiles to each breast cancer subtype.

7.1 Clustering With K-Means

Using the *kmeans* function in the R library, the *cluster_drugs.r* script is able to automatically cluster each M into k clusters. Using $2 < k < 8$, the clusters are then examined manually to determine the cluster P containing the point representing the current subtype disease. Using a common mean squared error formula, the other (drug) points within P are sorted by increasing distance to the sole disease point. In case of the disease point being the only point in P (common for $k > 3$), the script merges the closest cluster into P and proceeds as usual. Thus for each subtype i and value k, we obtain a subset of the drugs in M, ordered by closeness (criteria for repositioning) to the disease. Concatenated with the FDA approval statuses obtained in Section 6, this subset P is written to *k-means_subtype_s.csv* for each subtype $1 < s < 10$ and K-Means parameter $2 < k < 8$.