DETECTION OF CIRCULATORY TUMOR CELLS

Student Name: Ishmeet Kaur | Shubham Kumar
Roll Number: 2015042 | 2015098

BTP report submitted in partial fulfillment of the requirements
for the Degree of B.Tech. in Computer Science & Engineering
on July 7, 2018

BTP Track: Research Track

BTP Advisor
Dr. Debarka

Indraprastha Institute of Information Technology
New Delhi

Student's Declaration

I hereby declare that the work presented in the report entitled **Detection of Circulatory Tumor Cells** submitted by us for the partial fulfillment of the requirements for the degree of *Bachelor of Technology* in *Computer Science & Engineering* at Indraprastha Institute of Information Technology, Delhi, is an authentic record of my work carried out under guidance of **Dr. Debarka**. Due acknowledgements have been given in the report to all material used. This work has not been submitted anywhere else for the reward of any other degree.

..... **Place & Date: .....**
Ishmeet Kaur | Shubham Kumar

Certificate

This is to certify that the above statement made by the candidates is correct to the best of my knowledge.

..... **Place & Date: .....**
Dr. Debarka

Abstract

Detection of cancer has always been one of the most important and valued tasks in the history of medical science. In this project, we're trying to build some sort of a search engine where a person can feed their genomic data and, in the background we could detect the CTCs(circulatory tumor cells) in that and be able to present to the user, the possible types of Cancer he may be prone to have. Currently, we're in the first phase of training the deep learning neural network autoencoder model using h20 framework and are able to satisfactorily detect the CTCs.

Keywords: CTC: Circulating tumor cell, Neural networks, Auto-encoders, Anomaly detection

Acknowledgments

First of all, we would like to express our sincere gratitude to our advisor, Dr. Debarka, for his promptness and guidance, that always pointed us to the right direction whenever we needed the most, and for all the help and freedom he gave throughout the project. To our parents, Sarbans Singh, Gurpreet Kaur, Suresh Kumar and Sonia Kumar, our deepest heartfelt thank you, for always providing us with unfailing support and encouragement. We owe them what we are today and will be in the future.

Work Distribution

Ishmeet

Ch 2- Dataset (2.1: Dataset Acquisation, 2.2: Dataset Preprocessing and Cleaning)

Ch-3- Biological Concepts

Ch-4- Implementation(4.1, 4.2.1:Unsupervised neural network,4.3: Anomaly detection)

Shubham

Ch-2- Dataset (2.1: Dataset Acquisation, 2.3: Splitting the dataset)

Ch-3- Biological Concepts

Ch-4- Implementation(4.1, 4.2, 4.4: Supervised and unsupervised Autoencoders model)

Contents

Chapter 1

Introduction

It's no new news that every day, a lot of people lose their lives to cancer. Sadly, in many of these cases, the patient can't be treated well due to late detection of cancerous cells. This project aims at building an engine, in which, if one feeds their genomic data, the engine would tell them what all cancers they may be prone to have in the future, by detecting the highly rare CTCs(Circulatory tumor cells) in the genomic sample. It will not only lead to early detection but as well as a preventive measure for the people.

The detection of CTCs is limited by the rarity of these cells in the blood of the affected patients. CTCs have been known to play significant relevance in metastatic breast, prostate, and colorectal cancer. In this project, we are using the H2O python framework implementation of autoencoders and anomaly detection to separate the CTCS from the normal blood cells. Two approaches have been discussed. One being an unsupervised neural network autoencoders model, which uses the "bottleneck" technique, to detect these rare CTCs. The other approach uses a pre-trained supervised autoencoder model.

The supervised model performed better than the unsupervised one, and was satisfactorily able to differentiate the CTCs from the normal blood cells without misclassifying any of the normal blood cells.

Chapter 2

Dataset

2.1 Dataset acquisation

The dataset was acquired from 2 sources: one, from NCBI Genome Assembly and, from The Exome Aggregation Consortium (ExAC) publicly avaiable large scale human-genomic data.

htseq-count:

"Given a file with aligned sequencing reads and a list of genomic features, a common task is to count how many reads map to each feature. A feature is here an interval (i.e., a range of positions) on a chromosome or a union of such intervals." (1)

command:
htseq-count [options] [alignment-files] [gff-file]

here,

[alignment-files] : it is a file containing the aligned reads in BAM format.
[gff-file] contains the features in the GFF format("GFF is a standard file format for storing genomic features in a text file. GFF stands for Generic Feature Format. GFF files are plain text, 9 column, tab-delimited files. GFF databases also exist. They use a schema custom built to represent GFF data. GFF is frequently used in GMOD for data exchange and representation of genomic data."). We had used gencode.v28.annotation.gff3- It contains the comprehensive gene annotation on the reference chromosomes only. (source: https://www.gencodegenes.org/releases/current.html)
[options] : -f, -r

Final command used:
htseq-count -f bam -r pos 'xyz.bam' 'gencode.v28.annotation.gff3' ¿ result.txt

The output of the script is a table with counts for each feature, followed by special counters.

2.2 Preprocessing and Cleaning

Since, we acquired dataset from two different sources it was important for them to have the same features. The dataset from NCBI was already in csv format, the other one was first converted into csv files, with the commands written above. The first dataset matrix was translated, so that the rows had the cells and the columns had the features. Finally, we took an intersection of the two, resulting in a matrix of size 2719 * 17754, i.e. 17754 features, 2713 normal bood cells and 6 CTCs. It was then stored in a pandas dataframe.

Then, we did quantile normalisation with an already available python library on the dataframe. It is a technique for making two distributions identical in statistical properties. In this technique, first, both the test and the reference distributions are sorted. The highest entry in the test distribution then takes the value of the highest entry in the reference distribution until the test distribution is a perturbation of the reference distribution.

2.3 Splitting the dataset into train and test

The dataset was trained on the 2713 samples of normal blood cells. The 6 CTCs and the normal cells formed the test dataset.

Chapter 3

Biological concepts

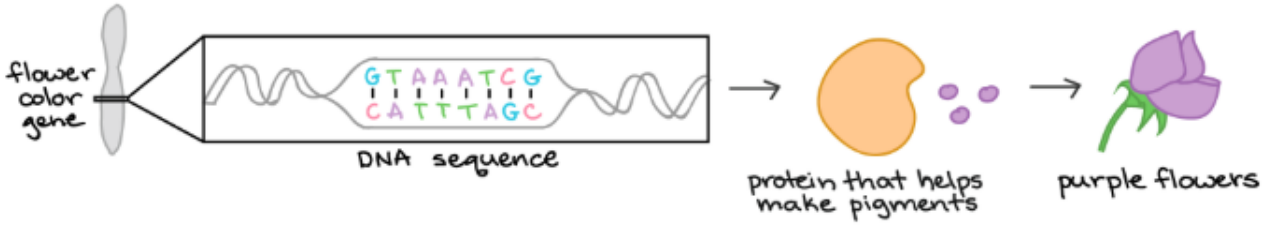
Before we get into the technical side of it, let's go through some of the basic concepts of biology useful for the project.

3.1 Gene Expression

When the DNA is transmitted from parents to children, it determines some of the major and minor characteristics in the children, like, eye color, hair color, etc.

3.1.1 Genes specify functional products (such as proteins)

A DNA strand is divided up into functional units called genes. The functional products of most known genes are proteins, or, more accurately, polypeptides- the chain of amino acids. Genes that specify polypeptides are called protein-coding genes.

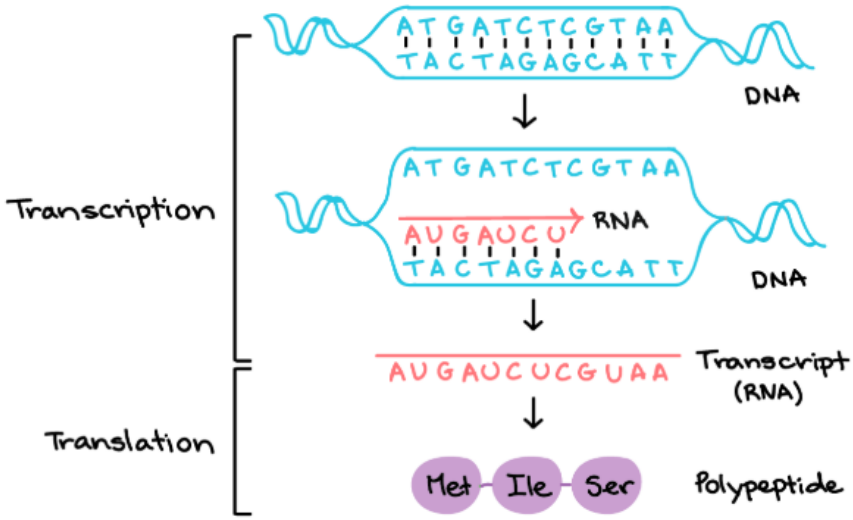


3.1.2 Transcription and Translation

DNA directs the construction of polypeptides through 2 main steps.

In transcription, the DNA sequence of a gene is copied to make an RNA molecule. This step is called transcription because it involves rewriting, or transcribing, the DNA sequence in a similar RNA "alphabet." In eukaryotes, the RNA molecule must undergo processing to become a mature messenger RNA (mRNA).

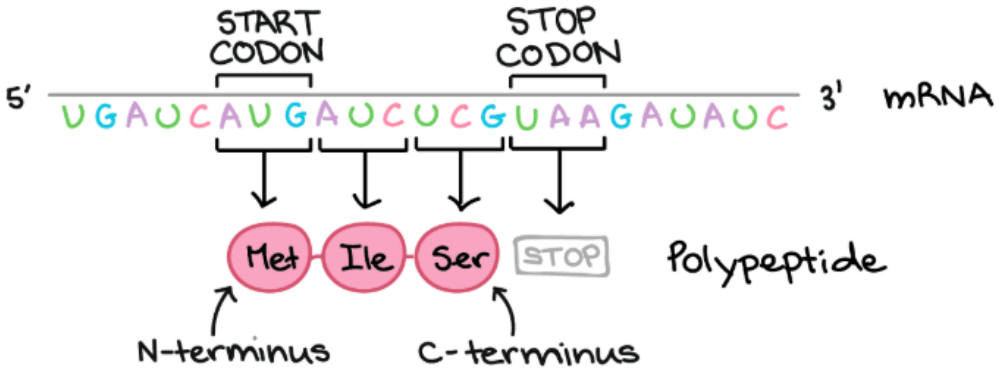
In translation, the sequence of the mRNA is decoded to specify the amino acid sequence of a polypeptide. The name translation reflects that the nucleotide sequence of the mRNA sequence must be translated into the completely different "language" of amino acids.



Thus, during expression of a protein-coding gene, information flows from DNA → RNA → Protein. The process of going from DNA to a functional product is known as gene expression.

3.2 The Genetic Code

mRNA is decoded by cells by reading their nucleotides in groups of three, called codons. Three "stop" codons mark the end of a protein and the "start" codon, AUG, marks the beginning of a protein. Codons in mRNA are read during translation, beginning with a start codon and continuing until a stop codon is reached. Codons are read from 5' to 3' , and they specify the order of amino acids in a protein from N-terminus to C-terminus.



The full set of relationships between codons and amino acids (or stop signals) is called the genetic code.

Chapter 4

Technological concepts

H2O is fast, scalable, open-source machine learning and deep learning for Smarter Applications.Companies like PayPal, Nielsen, Cisco use H2O to make faster and better predictions. Using in-memory compression, H2O handles billions of data rows in-memory. It includes interfaces for R, Python, Scala, Java, JSON, and Coffeescript/JavaScript, as well as a built-in web interface.H2O also implements best-in-class algorithms at scale, such as Random Forest, Gradient Boosting and Deep Learning.

4.1 Autoencoders

Deep learning can also be used for unsupervised feature learning or, more specifically, nonlinear dimensionality reduction. Consider a three-layer neural network with one hidden layer. If we treat our input data as labeled with the same input values, then the network is forced to learn the identity via a nonlinear, reduced representation of the original data. This type of algorithm is called a deep autoencoder; these models have been used extensively for unsupervised, layer-wise pre-training of supervised deep learning tasks along with its ability to discover anomalies in data.

4.2 Anomaly Detection

Suppose we have the autoencoder model discussed above, of three hidden layers. Given enough training data that resembles some underlying pattern, the network will train itself to easily learn the identity when confronted with that pattern. However, if some anomalous test point that does not match the learned pattern arrives, the autoencoder will likely have a high error in reconstructing this data, which indicates it is anomalous data.

4.3 Example

Let's create a simple autoencoder model.The dataset is an ECG time series of heartbeats and the goal is to determine which heartbeats are outliers. The training data (20 good heartbeats) and the

test data (training data with 3 bad heartbeats appended)

```
# Download and import ECG train and test data into the H2O cluster
train_ecg = h2o.import_file("http://h2o-public-test-data.s3.amazonaws.com
    /smalldata/anomaly/ecg_discord_train.csv")
test_ecg = h2o.import_file("http://h2o-public-test-data.s3.amazonaws.com/
    smalldata/anomaly/ecg_discord_test.csv")


# Train deep autoencoder learning model on "normal"
# training data, y ignored
anomaly_model = h2o.deeplearning(x=train_ecg.names,
                                 training_frame=train_ecg,
                                 activation="Tanh",
                                 autoencoder=True,
                                 hidden=[50,50,50],
                                 l1=1e-4,
                                 epochs=100)

# Compute reconstruction error with the Anomaly
# detection app (MSE between output layer and input layer)
recon_error = anomaly_model.anomaly(test_ecg)


# Note: Testing = Reconstructing the test dataset
test_recon = anomaly_model.predict(test_ecg)
test_recon
```

Chapter 5

Implementation and Results

5.1 Modelling

For modeling, we used Pythons h20 implementation with the h2o package. We used the autoencoders and anomaly detection from h20 to differentiate between the normal blood cells and circulatory tumor cells. We create h20 cluster using the init fucntion available, and initially assertions are disabled, which is used for debugging purposes, but sometimes messes up wit h the other things. Then we convert our panda dataframe to h20 dataframe, separately for train and test dataset.

5.2 Autoencoders

5.2.1 Unsupervised neural network model

Since the no. of CTCs and the no. of normal blood cells are in a ratio of 99:1, detecting them is like finding a needle in a haystack. so, we used the "bottleneck" training technique. The middle layer of the neural network is kept very small. So, the model will have to reduce the dimensionality of the input data. Here the size of the layers is : 10, 2 , 2 , so in this case, the dimension shall reduce to 2 nodes/dimensions.

The autoencoder model will then learn the patterns of the input data irrespective of given class labels. Here, it will learn, which blood cells are similar, i.e. can be classified as normal blood cells and which cells are outliers or anomalies, hence CTCs.

But, dimensionality reduction with our autoencoder model alone is not sufficient to identify outliers in this dataset. But we could use the reduced dimensionality representation of one of the hidden layers as features for model training, here, we use the 10 features from the third hidden layer.

For measuring model performance on test data, we need to convert the test data to the same reduced dimensions as the training data.

5.2.2 Pre-trained supervised model

We build a a feed-forward multilayer Deep Neural Network model, with tanh activation and 120 as the size of hidden layer and 100 epochs to begin with. Then the model is set to train and saved.

This model will now use the weights from the autoencoder for model fitting. We performed grid search for hyperparameter tuning.

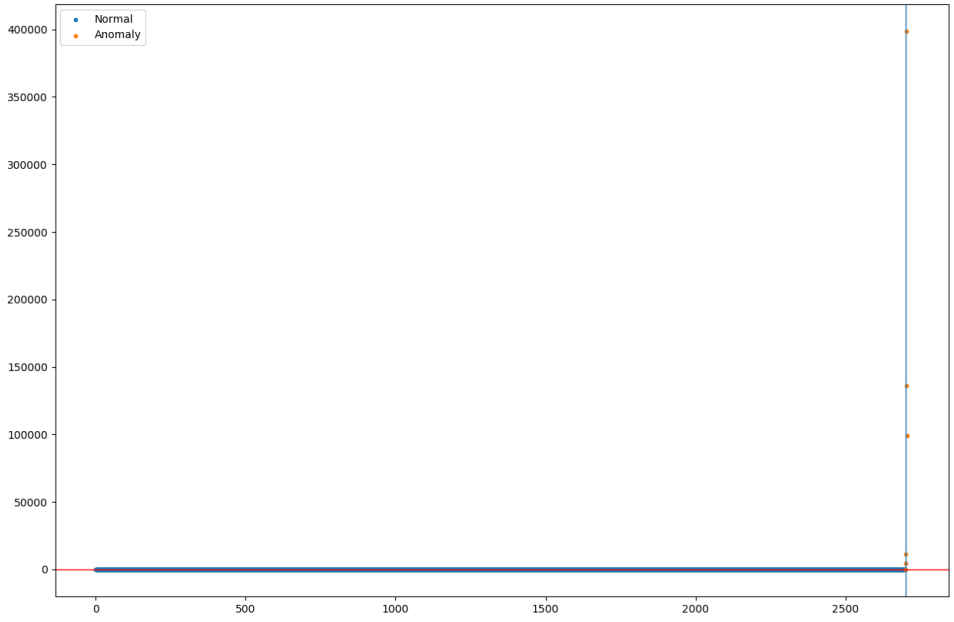
5.3 Anomaly detection

Then we use the anomaly detection from h20 package and get anomalous values on the test and train dataset separately. Based on the autoencoder model that was trained before, the input data will be reconstructed and for each instance, the mean squared error (MSE) between actual value and reconstruction is calculated.

5.4 Plots

Finally we converted our output to dataframe and scatter the plot with top whisker. The unsupervised model was able to detect all the CTCs but also labelled the normal blood cells as CTCs.

From the plot, we could see that the pre-trained model not only detected all the 7 CTCs but also did not declassify the normal blood cells.



Chapter 6

Conclusion and Future Work

As per the current dataset(2713 normal blood cells and 6 CTCs), the algorithm works perfectly fine in classifying the CTCs even after such a low no. of CTCs in the dataset without misclassifying the normal blood cells.

Currently, we need to expand our dataset and test the algorithm on bigger datasets and try applying engineering methods for improving its accuracy further. The ultimate the aim is to build the engine and integrate the algorithm in the backend.

Bibliography