# Predict the Effect of Genetic Variants to Enable Personalized Medicine

Guimin Dong
EECS Department
School of Engineering
Vanderbilt University
Nashville, Tennessee, 37203
Email: guimin.dong@vanderbilt.edu

***Abstract*—It is critical to determine how precise medicine is and, more concretely, how genetic testing is going to disrupt the way diseases like cancer are treated. Transitional detection and classification of clinical genetic mutations are mostly done by medical experts to read the medical literature and make decision about which kinds of this genetic mutation should belong. However, this cumbersome and fallible task makes such decision-making sometimes inconceivable. This paper discussed several machine learning algorithms to find the optimal classification in the supervised learning paradigm and finally used a hybrid model for classification.**

## I. Introduction

Although such promising expectation that advancing our understanding of cancer and designing more efficient, effective and precise treatments has been stimulated by genetic testing, the examination of genetic role is still slow because of significant amount of manual work. Memorial Sloan Kettering Cancer Center (MSKCC), in the past several years, has worked to create an expert-annotated precision oncology knowledge base. Thousand annotations of which genes are clinical actionable and which are not based on clinical literature have been complied. MSKCC launched a competition, accepted by the NIPS 2017 Competition Track, to develop classification models which analyze abstracts of medical articles and, based on their content accurately determine oncogenicity (4 classes) and mutation effect (9 classes) of the genes discussed in them.

Once sequenced, There are thousands of genetic mutation a cancer tumor can have. The challenge is to discriminate the mutations that contribute to tumor growth (drivers) from the neutral mutations (passengers), which are interpreted manually. A clinical pathologist review and classify every single genetic mutation based on evidence from clinical literature. For this competition MSKCC is making available an expert-annotated knowledge base where world-class researchers and oncologists have manually annotated thousands of mutations. Machine Learning algorithms are developed and discussed in this paper, which used the precision oncology knowledge base as a baseline, to automatically classify genetic variations.

## II. Data Set Description

The goal of this project is to develop machine learning algorithms to classify genetic mutations based on clinical evidence, where one of nine different classes a genetic mutation will be classified in. The input is the clinical evidence, implying that this task is nontrivial as it is still a challenge for clinical experts. Both, training and test, data sets are provided via two different files. One (training/test -variants) provides the information about the genetic mutations, whereas the other (training/test-text) provides the clinical evidence (text) that our human experts used to classify the genetic mutations. Both are linked via the ID field. Therefore the genetic mutation (row) with ID=15 in the file training-variants, was classified using the clinical evidence (text) from the row with ID=15 in the file training-text.

## III. Data Prepossessing and Feature Extraction

This section discusses the first step to train our machine learning model. The raw data input contains text of clinical evidence, which presented in data structure of string. The first part of this section is to discuss word2vec and term frequencyinverse document frequency (TF-IDF) method to represent the text input mapping from string to vector space, and the second part discusses of using truncated singular value decomposition (TSVD) method to generate a more efficient and effective representation of the raw input text with reduced dimension.

### *A. TF-IDF VS WORD2VEC*

In our mutation classification scenario, we have a word associates with a list of documents where the word exists in inverted indexing. When we apply TF-IDF, TF-IDF values of its document list will connect with the word. The similarity between two documents generally can be calculated by the term vectors, and relationship between the corresponding words from term vectors can also be calculated. However, such relationship if not semantically or syntactically related, implying that only a certain level of common occurrence in the textual units can be detected. Based on our clinical literature, the experts do not only determine the class of a certain gene mutation by counting the frequency of several batch of words, but understand the meaning and semantic reasoning among the information they can extract from literature. Thus, that lead us to consider Skim-gram model, which is a method of word2vector, to create better numerical representation of the

```
Variation : Y236S is not in the w2v vocabulary
Variation : Y236D is not in the w2v vocabulary
Y236C has a similar score with Gene TP53 of : 0.156223624065
Y234H has a similar score with Gene TP53 of : 0.104668291008
Y234C has a similar score with Gene TP53 of : 0.176433641758
Y220S has a similar score with Gene TP53 of : 0.13181626791
Y220C has a similar score with Gene TP53 of : 0.223609377548
Y163C has a similar score with Gene TP53 of : 0.00852609518266
Variation : V274F is not in the w2v vocabulary
V272M has a similar score with Gene TP53 of : 0.0606374069751
```

Fig. 1. Word similarity of TP53

clinical evidence, which can map text string to real value for each word.

Before, we step into Skip-gram, we first check whether there is null input in our input, and then find out that there are only few null input for each row, such as ID 1277, ID 2755, ID 1109 and so on. For the empty input, we can ignore the input and continue to work on out feature extraction. First and foremost, we extract text and target values and transform label to integer by using DataFrame in Pandas to contain the target series, where targets are the DataFrame Serie Name that contains all target values. Then we vectorize integer labels, with cleaned format by tokenization and string cleaning, and merge train and test data set, finally create a list of list from the text. However, such feature manipulation still admits unrelated word into our vector space by comparing feature frequency and word similarity, for example we can get the 10 most common variation from TP53 Gene and compare the frequency feature to the text similarity of each top variation to TP53, the result is showed in Fig.1.

We then create new features based on cancer synonym and related words by using the list of cancer similar words detected by word2vec, shown in the list: ['benign', 'malign', 'cell growth', 'abnormal cell growth', 'cell cycle', 'apoptosis', 'accumulation', 'spread', 'invade', 'lump', 'weight loss', 'oncological', 'survival', 'death', 'corruption', 'disease','malignancy', 'sickness', 'canker', 'carcinoma', 'big c', 'long illness', 'chemotherapy', 'radiotherapy','carcinogen', 'carcinogenic', 'carcinoma', 'cytology', 'cells grow', 'Hodgkin', 'leukemia', 'leukaemia', 'lymphoma', 'lymph nodes', 'malignancy', 'malignant', 'melanoma', 'metastasis', 'precancerous', 'sarcoma']. After that, we calculate the ratio as $\frac{frequency}{total\ number\ of\ words}$ for important bio-markers in the list: ['a-fetoprotein', 'AFP', 'hepatocellular', 'antigen-125', 'CA-125', 'human epididymis protein 4', 'HE4', 'ovarian cancer', 'thyroglobulin', 'Tg' , 'thyroid cancer', 'prostate specific antigen', 'PSA', 'prostate cancer', 'carcinogenic embryonic antigen', 'CEA', 'pancreatic cancer', 'CA15-3/CA27-29', 'HER2/neu', 'breast cancer.6'].

### B. Truncated SVD (TSVD)

To summarize singular value decomposition, let $\boldsymbol{A}$ be decomposed into the tree matrices $\boldsymbol{U}$, $\boldsymbol{\Sigma}$ and $\boldsymbol{V}$:

$$\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^{\boldsymbol{T}} \tag{1}$$

where the left and right singular matrices $\boldsymbol{U} \in \boldsymbol{R}^{m\times m}$ and $\boldsymbol{V} \in \boldsymbol{R}^{n\times n}$ are orthogonal, and where matrix $\boldsymbol{\Sigma} \in \boldsymbol{R}^{m\times n}$ has diagonal form:

$$\boldsymbol{\Sigma} = diag(\sigma_1, \sigma_2, ..., \sigma_n) \tag{2}$$

The diagonal elements $\sigma_i$ of $\boldsymbol{\Sigma}$ are the singular values of $\boldsymbol{A}$, and they are ordered such that:

$$\Sigma_1 \geq \Sigma_2 \geq ... \geq \Sigma_r > \Sigma_{r+1} = ... = \Sigma_n = 0 \tag{3}$$

where $r = rank(\boldsymbol{A})$.

The basic idea of TSVD is to impose the additional requirement on the solution that is norm be small, thus can ignore the trivial contributions from the errors of the right-hand side. In the case of TSVD, this is achieved by neglect of the components of the solution corresponding to the smallest singular values. Thus, the TSVD of $\boldsymbol{A}$ is defined as the rank-$k$ matrix

$$\boldsymbol{A_k} = \boldsymbol{U}\boldsymbol{\Sigma_k}\boldsymbol{V^T},\ \boldsymbol{\Sigma_k} = diag(\sigma_1, ..., \sigma_k, 0, ..., 0) \in \boldsymbol{R^{m\times n}} \tag{4}$$

By using TSVD technique, which we select the first 60 components, we can use such matrix as our input to train our machine learning models.

## IV. Model Selection

### A. Multi-Layer Perceptron (MLP)

As this project is for Kaggle competition, it is too trivial to model a classifier by assuming that memberships of gene mutation are linearly classifiable or to use logistic regression for classification purpose. Thus, we start from an Artificial Neural Network model, Multi-Layer Perceptron, which is a well-known method for non-linear classifier.

*1) MLP Architecture:* MPL aims to learn a function $f(\bullet) : \boldsymbol{R^m} \rightarrow \boldsymbol{R^o}$ by training on a dataset which in the paradigm of supervised learning algorithm. In function that MLP will learn, $m$ is the number of dimensions for input and $o$ is the number of dimensions for output. For the purpose of this project, $m = 60$ and $n = 9$, as we have 60 features and 9 classes in our dataset. For the set of input, $\boldsymbol{X}$ and the output $\boldsymbol{Y}$, MLP will learn non-linear function approximation for $y_i$. There must be at least 3 layers for MLP, input layer, hidden layer and output layer which are constituted by neurons as the dimension of layer, and each layer is completed connected by directed edges to send signals from previous layer to the next, as shown in Fig.2 an architecture of MLP.

*2) MLP Dynamics:* The first layer involves $M$ linear combinations of the $d$-dimension inputs:

$$b_j = \sum_{i=0}^{d} w_{ji}^{(1)} x_i,\ j = 1, 2, ..., M. \tag{5}$$

where $b_j$ acts as activation function, and $w_{ji}^{(1)}$ is weight. Then this activation function send signal from the input to the hidden layer, transformed by $sigmoid$ function:

$$z_j = \frac{1}{1 + exp(-b_j)} \tag{6}$$

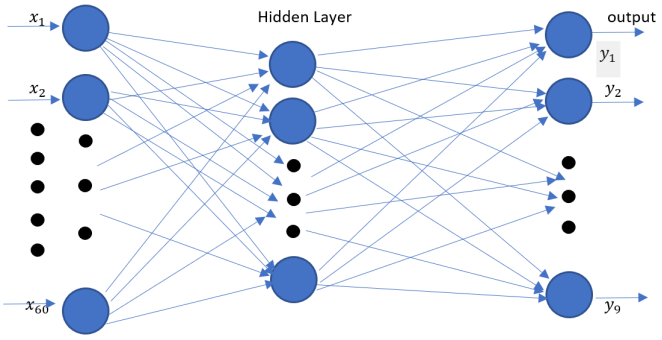


Fig. 2. MLP Architecture

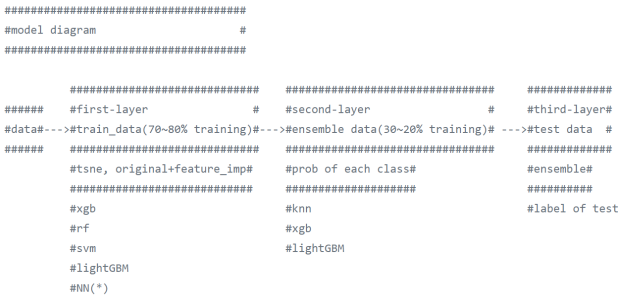


Fig. 3. Ensemble Learning Architecture

And then, for the hidden layer, the outputs of hidden units are linear combination of the outputs from the first layer, activated by function (7):

$$a_h = \sum_{j=0}^{M} w_{hj}^{(2)} z_j, \ \ h = 1, 2, ..., H. \tag{7}$$

Then send signal to the output layer with $softmax$ activation function:

$$y_k = \frac{exp(a_k)}{\sigma_{l=1}^{K} exp(a_l)} \tag{8}$$

for our muli-classification purpose.

To train our MLP model, we set up the cost function, $cross\ entropy\ error$:

$$CE_{loss} = -\sum_{i=1}^{9} (t_i log(y_i)) + (1 - t_i) log(1 - y_i) \tag{9}$$

where $t_i$ is the target vector. Then we use stochastic gradient descent to evaluate the derivative of $\frac{\partial CE_{loss}}{\partial w_{hj}^{(2)}}$ and $\frac{\partial CE_{loss}}{\partial w_{ji}^{(1)}}$, which can be performed by back propagation algorithm.

*3) MLP Implementation:* Instead of using $sigmoids$ activation function, we use $Rectified\ Linear\ Units$(ReLUs)

$$f(x) = max(x, 0) \tag{10}$$

to transform the signal from the first layer, which will treat the input as the same one and if the input is less than 0, set the input as equal to 0. We set up 60 units at the first layer corresponding to the input dimension, 80 units for the hidden layer empirically, which should be greater than the number of input neuron, and 9 units for the output layer corresponding to the number of classes. To keep the activation function operates as we expected, we initialize the weights to be small around the origin which are randomly generated by normal distribution with 0 mean and 1 standard deviation. In training the model, we set the initial learning rate as 0.01, and update the learning rate with $progressive\ decay$ as $10^{-6}$: initial learning rate $\theta = \theta_0$, learning rate decay $\theta_d$, at each iteration $s$: $\theta(s) = \frac{\theta_0}{1+s\times\theta_d}$. And finally set up the momentum for SGD as 0.9

### B. Ensemble Learning for Multi-Classification

There many successful machine learning algorithms which can achieve both efficient and robust estimation. However, each algorithm has their own virtue and vice. For example, SVM is not sensitive to multi-classification, and logistic regression is too rigid to its linearity assumption of the $log\ likelyhood\ ratio$ . Thus it is brilliant to combine the advantages of different classification models to make more accurate prediction.

*1) Ensemble Learning Architecture and Dynamics:* The architecture of ensemble learning is pretty similar to MLP algorithm. In the first layer, several classification models can be trained and cross-validated. And each model represented as a node in the first layer will output a vector of predicted labels, as shown in Fig. 3, and send the predicated label signals to the second layer to train a smaller bunch of classification models, which then send out the predicted label signal to the third layer. Finally, in the third layer, we can use the average method to average the predictions to achieve our final label prediction.

*2) Ensemble Learning Implementation:* In the first layer of ensemble learning architecture, we train five classification models including $xgb$, $random\ forest$, $support\ vector\ machine$ with radical kernel, $light\ GBM$, and $knn$. Then after cross-validation for each models in the first layer, predicted label signal sent out to the second layer. In the second layer, we train $knn$, $xgb$, and $light\ GBM$ models to prevent overfitting. And finally, for the third layer to calculate the average for each predication from the labels sent out from the second layer.

## V. Results and Discussion

For each ID in the final test set provided in Kaggle, we predict a probability for each of the different classes a genetic mutation can be classified on. The result is evaluated by $multi\ class\ log\ loss$

$$logloss = -\frac{1}{N} \sum_{i=1}^{N} \sum_{j=1}^{M} y_{i,j} log(p_{i,j}) \tag{11}$$

where, N is the number of observations, M is the number of class labels, $log$ is the natural logarithm, $y_{i,j}$ is 1 if observation

$i$ is in class $j$ and 0 otherwise, and $p_{i,j}$ is the predicted probability that observation $i$ is in class $j$. By using $MLP$ the private score is 3.56452, the public score is 1.55374, and by using ensemble learning method the private score is 3.07373 and the public score is 0.054094, indicating the ensemble learning advanced $MLP$ in this genetic mutation classification.

The feature engineering section can be improved by implementing some state-of-art methods. And it is critical to have an effective and efficient low dimensional representation of the original data matrix for training learning algorithms. Furthermore, for this specific context environment, the clinical literature, it is far-fetched to use the $word2vec$ to process the raw input. The result can be improved by using professional clinical literature of word embedding in the future.Besides, $CNN$ and $RNN$ model did not be discussed in this project, which I believe that these two models can generate interesting solution.

## REFERENCES


[1] Vapnik, V., *Statistical Learning Theory* , New York: Wiley, 1998.

[2] Speed, T.(ed.), *Statistical Analysis of Gene Expression Microarray Data*, Chapman and Hall, London, 2000

[3] Schapire, R., The boosting approach to machine learning: an overview, in D. Denison, M. Hansen, C. Holmes, B. Mallick and B. Yu (eds), *MSRI workshop on Nonlinear Estimation and Classification*,Springer, New York, 2002

[4] Ripley, B. D., *Pattern Recognition and Neural Networks*, Cambridge University Press, 1996.

[5] Barrett, J., Cairns, D., Application of the random forest classification method to peaks detected from mass spectrometric proteomic profiles of cancer patients and controls, *Statistical Applications in Genetics and Molecular Biology*, 7(2), 2008

[6] Rosenblatt, F., The perceptron: a probabilistic model for information storage and organization in the brain*Psychological Review*, 65: 386408, 1958.

[7] Ethem A., *Introduction to Machine Learning (2nd)*, The MIT Press Cambridge, Massachusetts London, England, 2009

[8] Trevor H., Robert T, Jerome F., *The Elements of Statistical Learning: Data Mining, Inference, and Predication*, Springer, New York, 2009

[9] Christopher M. B., *Pattern Recognition and Machine learning*, Springer, New York, 2006

[10] Christopher D. M., Hinrich S., *Foundations of Statistical Natural Language Processing*, The MIT Press Cambridge, Massachusetts London, England, 1999