Hellenic Mediterranean University

School of Engineering

Department of Electrical and Computer Engineering

A Thesis
Bachelor of Science

Κατηγορηματική επισημείωση με τη χρήση διανυσματικών παραστάσεων και βαθέων νευρωνικών δικτύων
(Categorical Embedding with Deep Learning)

Giannakos Iakovos
ΤΠ4127

Suppervisors: Dr. Tsiknakis Manolis, Dr. Koumakis Lefteris and Dr. Marias Konstantinos.

Abstract

The study conducted in the framework of the dissertation on "Categorical embedding with deep learning". To clarify, the purpose of the dissertation is to study and implement a word-embedding neural network for genomic data which is a network consisting of three levels namely the input level, the hidden level and the output level. All these levels are interconnected with different forces (weights) which are also called word-embedding.

The selected architecture of the neural network falls in the Natural Language Processing (NLP) category. NLP is a research field that investigates how a computer can control and extract knowledge from text or dialogue into a natural language. The model implemented in this dissertation is the Continues Bags of Words (CBOW), a model that accepts as input a set of number boxes (contexts) which are the number of words corresponding to a text. Each context corresponds to several words defined by the developer, has a target context and a table with the difference of the words in a text that correspond to that context. The network is trained with the assumption that each context is close to the words that are the target. The aim is to train the CBOW neural network and to form word embedding using as input known mutations of a human.

Before we get to the training point, the network requires some data as input. Our data comes from the human genome using the Ensembl Variant Effect Predictor (VEP). Our main objective is to get all the human mutations (about 80 million mutations) and train a model that will handle each mutation as a word and each disease as the context. VEP is a tool for annotating, evaluating and prioritizing genomic mutations, even in non-coding areas. The VEP predicts the effects of sequence mutations on transcripts, protein products, regulatory regions, and binding patterns, utilizing the high quality, wide scope, and comprehensive design of Ensemble databases with high accuracy. In the next, we pass the variants/mutations to a python script where we select input features based on specific criteria described in chapter Experiment 1 (sub section Data) and Experiment 2 (sub section Data). After selecting the data, we form the context list with the data and a target context for each single-nucleotide polymorphism (SNP) variant. Then the CBOW model is trained with the variants contexts mentioned above and after some epochs the embedding (weights) that are between the first level and the hidden are formed. We extract these weights from the network and pass them to the Principal Component Analysis (PCA) to visualize it as a scatter plot. PCA, is a dimensionality-reduction method that is often used to reduce the dimensionality of large data sets, by transforming a large set of variables into a smaller one that still contains most of the information in the large set. Finally, cosine similarities were used. Cosine similarity is a measurement in data retrieval. The application of this measurement can be applied to two corpuses (paragraph, sentence and the whole corpus). If the similarity score is high between two corpus term vector and the query vector, the greater relevance of text and query. Once we have taken a SNP as a sample and passed it through cosine similarity we can find other SNP's close to this that we expect to be more similar so there is a possibility that this mutation will affect our sample.

We applied this methodology to three experiments. The first one was for the representation and clustering of human chromosome 22 variants. In this experiment we attempt to find relevance between random SNPs and verify them. Due to the large amount of chromosome data and processing time it was hard to have the best possible results. So we moved to the second and third experiment with less data targeted to a disease, specific in cancer variant and possible cancer variants. The results of the model are promising and we believe that such a methodology could be used in the genomics era.

Περίληψη

Η παρούσα μελέτη διεξήχθη στα πλαίσια της πτυχιακής εργασίας με θέμα «Κατηγορηματική επισημείωση με τη χρήση διανυσματικών παραστάσεων και βαθέων νευρωνικών δικτύων». Ο σκοπός της πτυχιακής εργασίας είναι να μελετήσει και να εφαρμόσει ένα word-embedding νευρωνικό δίκτυο για γονιδιωματικά δεδομένα όπου θα είναι ένα δίκτυο που αποτελείται από τρία επίπεδα, δηλαδή το επίπεδο εισόδου, το κρυφό επίπεδο και το επίπεδο εξόδου. Όλα αυτά τα επίπεδα διασυνδέονται με διαφορετικές δυνάμεις (βάρη) που ονομάζονται επίσης word-embedding. Αυτή η εργασία χρησιμοποιεί βάρη από το επίπεδο εισόδου έως το κρυφό επίπεδο.

Η επιλεγμένη αρχιτεκτονική του νευρικού δικτύου εμπίπτει στην κατηγορία Natural Language Processing (NLP). Το NLP είναι ένα πεδίο έρευνας που διερευνά τον τρόπο με τον οποίο ένας υπολογιστής μπορεί να ελέγξει και να εξαγάγει γνώσεις από ένα κείμενο ή διάλογο σε μια φυσική γλώσσα. Το μοντέλο που εφαρμόζεται σε αυτήν την εργασία είναι το Continues Bags of Words (CBOW), ένα μοντέλο που δέχεται ως εισαγωγή ένα σύνολο αριθμητικών κουτιών (contexts) όπου είναι οι αριθμητικές λέξεις που αντιστοιχούν σε ένα κείμενο. Κάθε context αντιστοιχεί σε αρκετές λέξεις που ορίζονται από τον προγραμματιστή, έχει ένα target context και έναν πίνακα με τη διαφορά των λέξεων σε ένα κείμενο που αντιστοιχεί σε αυτό το context. Το δίκτυο εκπαιδεύεται με την υπόθεση ότι κάθε context είναι κοντά στις λέξεις που είναι ο στόχος. Ο στόχος είναι να εκπαιδεύσει το νευρωνικό δίκτυο CBOW και να διαμορφώσει το word embedding χρησιμοποιώντας ως μεταλλάξεις εισόδου των homo sapiens.

Πριν φτάσουμε στο σημείο εκπαίδευσης, το δίκτυο απαιτεί ορισμένα δεδομένα ως είσοδο. Τα δεδομένα μας προέρχονται από το ανθρώπινο γονιδίωμα χρησιμοποιώντας το Ensembl Variant Effect Predictor (VEP). Ο κύριος στόχος μας είναι να πάρουμε όλες τις ανθρώπινες μεταλλάξεις (περίπου 80 εκατομμύρια μεταλλάξεις) και να εκπαιδεύσουμε ένα μοντέλο που θα χειρίζεται κάθε μετάλλαξη ως λέξη και κάθε ασθένεια ως το context. Το VEP είναι μια ισχυρή εργαλειοθήκη για την αξιολόγηση, τον σχολιασμό και την ιεράρχηση των γονιδιωματικών παραλλαγών, ακόμη και σε περιοχές που δεν κωδικοποιούν. Το VEP προβλέπει με ακρίβεια τα αποτελέσματα των παραλλαγών ακολουθίας σε μεταγραφές, πρωτεϊνικά προϊόντα, ρυθμιστικές περιοχές και δεσμευτικά μοτίβα, χρησιμοποιώντας την υψηλή ποιότητα και το ευρύ πεδίο για τον ολοκληρωμένο σχεδιασμό βάσεων δεδομένων Ensemble.

Στην συνέχεια, μεταβιβάζουμε τις παραλλαγές / μεταλλάξεις σε ένα script python όπου επιλέγουμε τα χαρακτηριστικά εισαγωγής βάση συγκεκριμένων κριτηρίων που περιγράφονται στο κεφάλαιο Experiment 1 (υποενότητα Data), Experiment 2 (υποενότητα Data) και Experiment 3 (υποενότητα Data). Αφού επιλέξουμε τα δεδομένα, διαμορφώνουμε τη context λίστα με τα δεδομένα και ένα target context για κάθε παραλλαγή πολυμορφισμού μονού νουκλεοτιδίου (SNP). Στη συνέχεια, το μοντέλο CBOW εκπαιδεύεται με τα variants contexts που αναφέρονται παραπάνω και μετά από μερικά epochs σχηματίζονται τα βάρη που βρίσκονται μεταξύ του πρώτου επιπέδου και του κρυφού επιπέδου. Εξάγουμε αυτά τα βάρη από το δίκτυο και τα μεταφέρουμε στην Ανάλυση Κύριου Συστατικού (Principal Component Analysis - PCA) για να το απεικονίσουμε με ένα διάγραμμα διασποράς. Το PCA είναι μια τεχνική που χρησιμοποιεί εξελιγμένες μαθηματικές αρχές για τη μετατροπή αρκετών δυνητικά συσχετισμένων μεταβλητών σε μικρότερο αριθμό μεταβλητών που ονομάζονται κύρια συστατικά. Εν συντομία βρίσκει μέχρι τρεις διαστάσεις από δεδομένα με N (N>3) διαστάσεις.

Τέλος, χρησιμοποιήθηκε ομοιότητες με συνημίτονα (Cosine Similarity). Η ομοιότητα συνημίτονων είναι μια ευρέως χρησιμοποιούμενη μέτρηση στην ανάκτηση πληροφοριών και σε σχετικές μελέτες. Η εφαρμογή αυτής της μέτρησης μπορεί να εφαρμοστεί σε δύο κείμενα (πρόταση, παράγραφος ή ολόκληρο το έγγραφο). Όσο υψηλότερη είναι η βαθμολογία

ομοιότητας μεταξύ του διανύσματος όρου εγγράφου και του διανύσματος όρου ερωτήματος, τόσο μεγαλύτερη είναι η συνάφεια μεταξύ του εγγράφου και του ερωτήματος. Μόλις πάρουμε ένα SNP ως δείγμα και το περάσουμε μέσω ομοιότητας συνημίτονου, μπορούμε να βρούμε άλλα SNP κοντά σε αυτό που αναμένουμε να είναι πιο όμοια, έτσι υπάρχει πιθανότητα αυτή η μετάλλαξη να επηρεάσει το δείγμα μας.

Εφαρμόσαμε αυτήν τη μεθοδολογία σε τρία πειράματα. Το πρώτο ήταν για την αναπαράσταση και την ομαδοποίηση των παραλλαγών του ανθρώπινου χρωμοσώματος 22. Σε αυτό το πείραμα προσπαθούμε να βρούμε συσχέτιση μεταξύ τυχαίων SNP και να τα επαληθεύσουμε. Λόγω του μεγάλου όγκου δεδομένων και του χρόνου επεξεργασίας ήταν δύσκολο να έχουμε μια την καλύτερη εικόνα των αποτελεσμάτων. Έτσι, προχωρήσαμε στο δεύτερο και τρίτο πείραμα με λιγότερα δεδομένα που στοχεύουν σε μια ασθένεια, συγκεκριμένα σε μεταλλάξεις καρκίνου και πιθανές μεταλλάξεις καρκίνου. Τα αποτελέσματα του μοντέλου είναι πολλά υποσχόμενα και πιστεύουμε ότι μια τέτοια μεθοδολογία θα μπορούσε να χρησιμοποιηθεί στην περιοχή της γονιδιωματικής.

Table of Contents

Tables and Figures

1 Introduction

This dissertation investigates the problem of creating an NLP word-embedding from variants data obtained by the Variant Effect Predictor (VEP) [1]. Nowadays, deep neural networks, aka deep learning, gain momentum in many research areas especially in image processing [2][3], even though one can identify many efforts in the fields of data analysis using deep learning. In this thesis we explore the possibility to use a deep learning methodology for the classification of diseases based on reported mutations. The rationale behind this effort comes from the text mining area. Deep learning methods have proven to be very effective in predicting the next word of a sequence of words [4] or classifying documents. Using the same principles as document classification we explore the possibility to use the variants of a person in order to classify/identify possible diseases. Based on our theory, the variants are handled as words and we try to train a deep learning model that can classify persons to diseases based only on their variants.

A mutation is a change in our DNA, those errors can be happening during copying DNA or as a result of environmental factors such a UV light. Our DNA will undergo sequence changes in the genome bases A, C, G and T over a lifetime. The proteins that are produced are therefore changed and that can be just a good deal or bad. If those errors where not fixed by the time, mutations can occur during DNA replication [5]. For instance, a mutation in the gene that is responsible for the construction of a protein called hemoglobin (Hb) causes the sickle cell anemia. This produces an irregular, stiff, sickle form of the red blood cells. However, having this mutation in African populations also protects against malaria [6]. We need a solution to figure, identify and categorize some correlations of mutations from a patient like the example above, in order to study them and draw new conclusions and find their similarities as well as how one mutation can affect another.

Our approach differs from the majority of other tools in the sense that we are using deep learning algorithms. For the realization of our approach we implemented the following steps. VEP data was processed and we selected some of the available fields. The data (SNPs that have impacts to a target SNP) is entered in a neural model type Continuous Bag of Words (CBOW) and after its training the word embedding - vectors were formed from the hidden level of the network. The word embedding exported first from the neural and then we imported it into the Principal Component Analysis (PCA) where three Principal Components formed, i.e. we will represent the variants in 2 or even 3 dimensions using a scatter plot. The last step was to use cosine similarity to find the most similar ones and see if the results are valid due to the similarity of the variants using a database e.g., the SNP database of NCBI.

1.1 Previous Research

This sub-section provides the literature review in the field of variant calling using deep learning. A mini review of deep learning methodologies for bioinformatics can be found here [2].

1.1.1 Functional interpretation of genetic variants using deep learning predicts impact on chromatin accessibility and histone modification.

The difficulty of understanding genetic variants' functional consequences currently limits the identification and implementation of personalized medicine of the functional variants underlying risk of disease. The functional consequences of disease related protein coding variants are more and more routine. Nevertheless, the large majority of risk variants are non-coding, and the prediction of functionality and priority variants for functional validation remains a major challenge. Here, the authors (Hoffman G., Bendl J., Girdhar K., Schadt E., Roussos P.) developed a deep learning template in order to find specific signals to be predicted with the input of the DNA sequence from four epigenetic experiments. In view of the expected epigenetic signals of the DNA sequence and the alternative alleles at a given site, the authors provided a score of the predicted epigenetic consequences for 438 million variants found in previous sequencing projects. These impacts are test-specific, are predictive of binding the transcription factor and are enriched for genetic expression and variants associated with disease-related risk. Nucleotide-level functional effects score for non-coding variants will refine the mechanism of functional variants, categorize new risk variants and prioritize the subsequent experiments. [7]

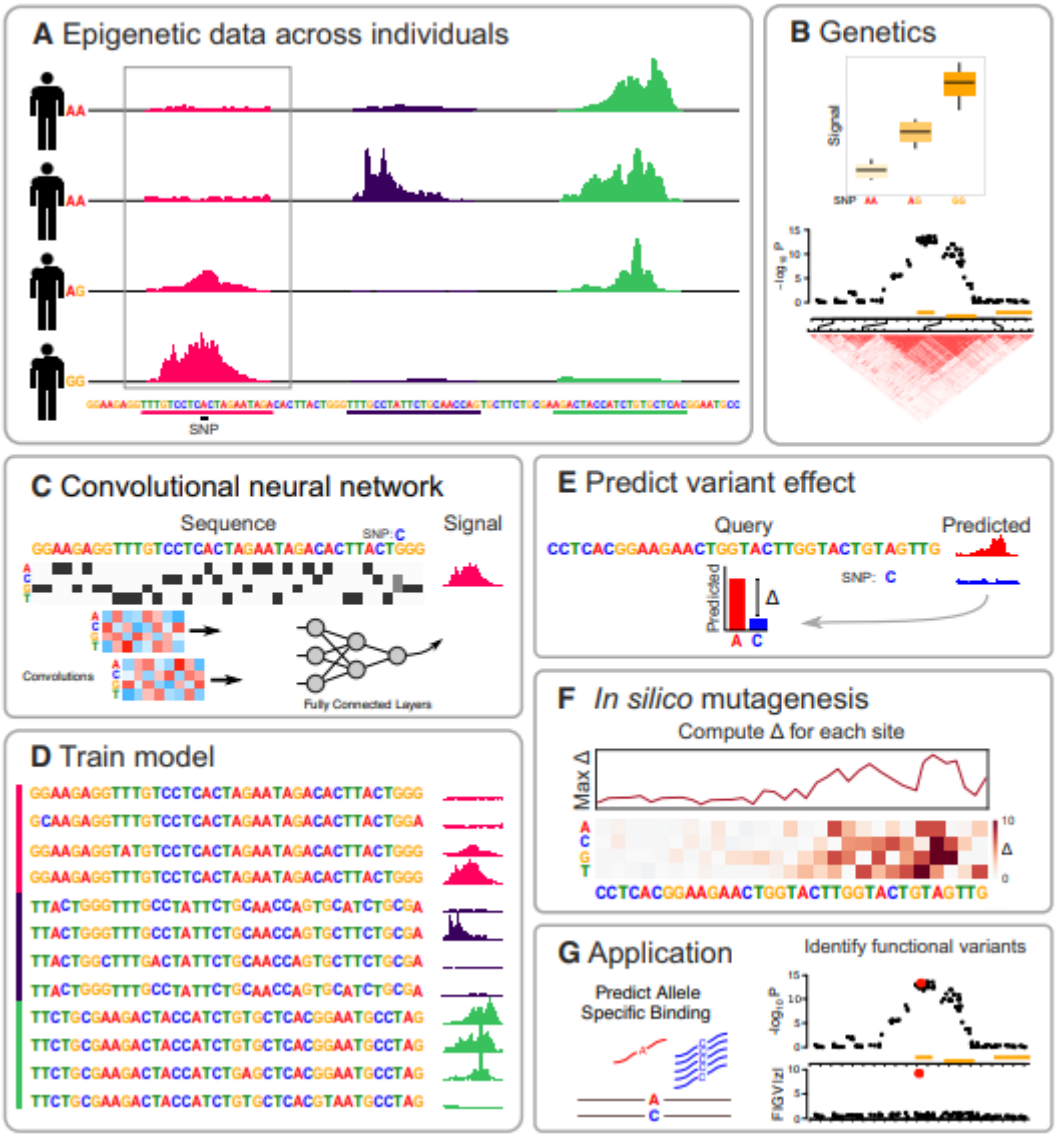


Figure 1.1: Computational workflow for Deep Functional Interpretation of Genetic Variants (DeepFIGV).

Figure1.1 highlights: (A) Epigenetic quantitative signal (i.e. ChIP-seq, DNase-seq) for different individuals and for genomic regions.

(B) Normal genetic analysis stratifies the quantitative signal by allelic at a given SNP, however unbalance in linkage complicates the functional variant interpretation.

(C) DeepFIGV encodes a sequence of DNA as a 'Picture' matrix usually of zeros including 1 (i.e. a Dark Box) showing the presence in that location of a specific nucleotide. Each allele is encoded as 0.5 heterozygous SNPs. Local matrix operations containing parameter values learned from the data are Convolutions. A neural network predicts the epigenetic signal of the DNA sequence using the convolutions.

(D) The computational model training ties a wide range of DNA sequences to the epigenetic signal of each region.

(E) For a query sequence with the reference and the alternative allele, the epigenetic signal is calculated. The difference between the predicted signal (for example delta) values shows that the variant has a predicted effect.

(F) The delta value for any potential nuclear substitution is evaluated in silico mutagenesis.

(G) DeepFIGV delta was used to predict and classify the functional variants of candidates through allele complex binding of transcription factors.

1.1.2 A universal sNP and small-indel variant caller using deep neural networks

Despite significant advancements in sequence technology, genetic variants from billions of short misleading sequence readings in a single genome remain difficult to call reliably. Authors demonstrate in this research that a deep convolutional neural network can call for genetic variation in aligned next generation read results by learning statistical associations between pileup images about suppositive variant or true genotype calls. The software, known as Deep Variant, outperforms the instruments and the learning paradigm extends to all genome structures and mammalian species, allowing the human to extract earth-reality data from non-human sequencing projects. Authors also show that the benefits of more automated and common variable-calling strategies are illustrated with a number of sequencing technologies and experimental designs. This involves deep genomes from the 10X genomic and ion ampliseq exomes. [8]

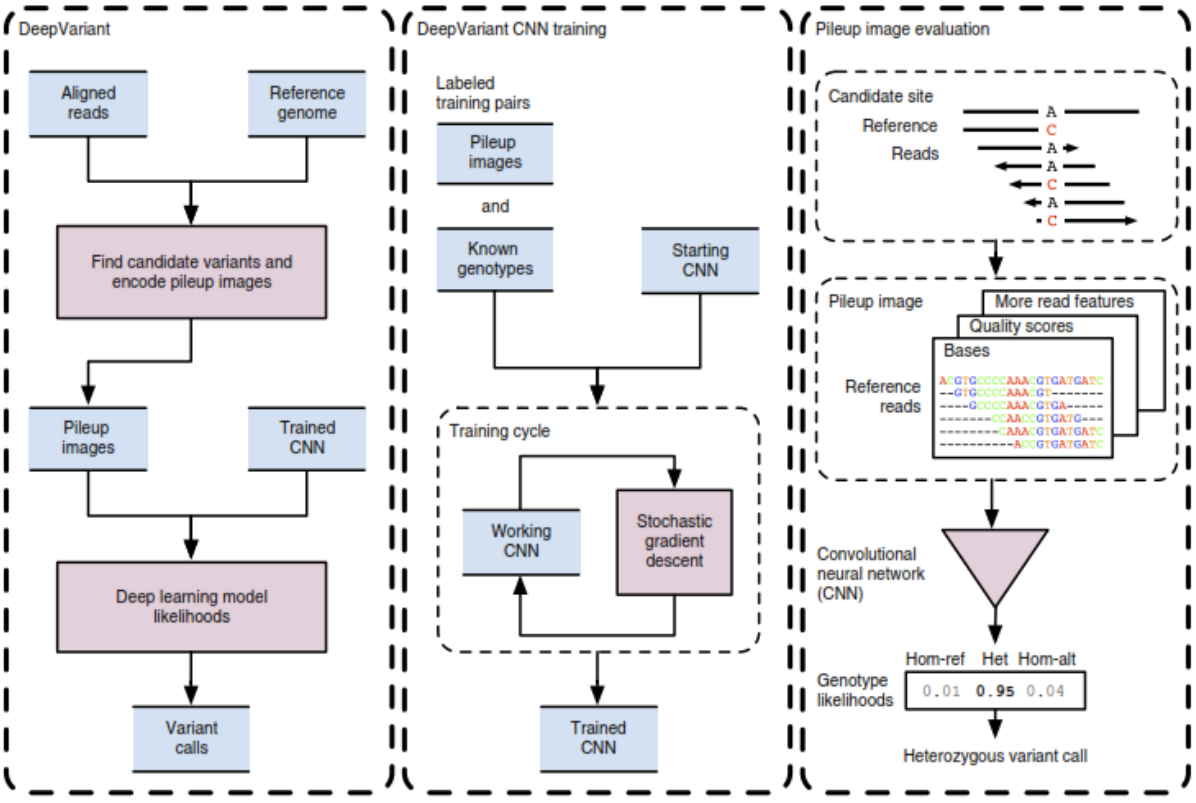


Figure 1.2: DeepVariant workflow overview.

Before DeepVariant, NGS reads are first aligned to a reference genome and cleaned up with duplicate marking and, optionally, local assembly. NGS reading is first matched to a reference

genome before DeepVariant and cleaned up by duplicate labeling and optionally local assembly. The left box: first the aligned reads for various sites of the reference genome are scanned. For each site of the candidate variant the read and reference data are encoded as an image. The genotype probabilities of each site are determined by a trained CNN. Whether a heterozygous or homozygous non-reference is most likely a variant is emitted. Middle Box: the CNN training reuses the DeepVariant machinery for generating pile-up images for a known genotype sample. These labeled picture + genotype pairs, together with an original CNN, which can be a random model, CNN trained for other image classification tests, or an earlier DeepVariant model, optimize the CNN parameters with the aid of a stochastic gradient descent algorithm to improve genotype prediction accuracies. The final trained model is frozen and can then be used in the variant calling after the completion of a maximum number of cycles, times or the output of the model has converged. The right box: the bases and reading, quality values, and other read characteristics are encoded in a pile-up image red-green-blue (RGB) in a candidate version. This image encoded is given by the CNN to measure the probability genotype of the three homozygotic (hom ref), heterozygotic (het) or homozygotic alternative diploid genotype states (hom alt). A heterozygous variant name will be given in this case, since the most likely genotype is "het." Blue boxes are data in all panels and red boxes are procedures. Online approaches provide descriptions of all procedures.

1.1.3 Predicting effects of noncoding variants with deep learning–based sequence model.

It is a major challenge in human genetics to recognize functional effects of noncoding variants. A deep learning-based algorithmic system, DeepSEA (http:/deepsea.princeton.edu/), implemented in order to predict the non-coding variants effects from a sequence, where learns a regulatory sequence code from large-scale chromatin-profiling results, enabling prediction of chromatin effects of single-nucleotide sensitivity sequence alterations. This ability was also used to enhance the prioritization of functional variants, including quantitative trait loci (eQTLs) expression and variants associated with disease. [9]

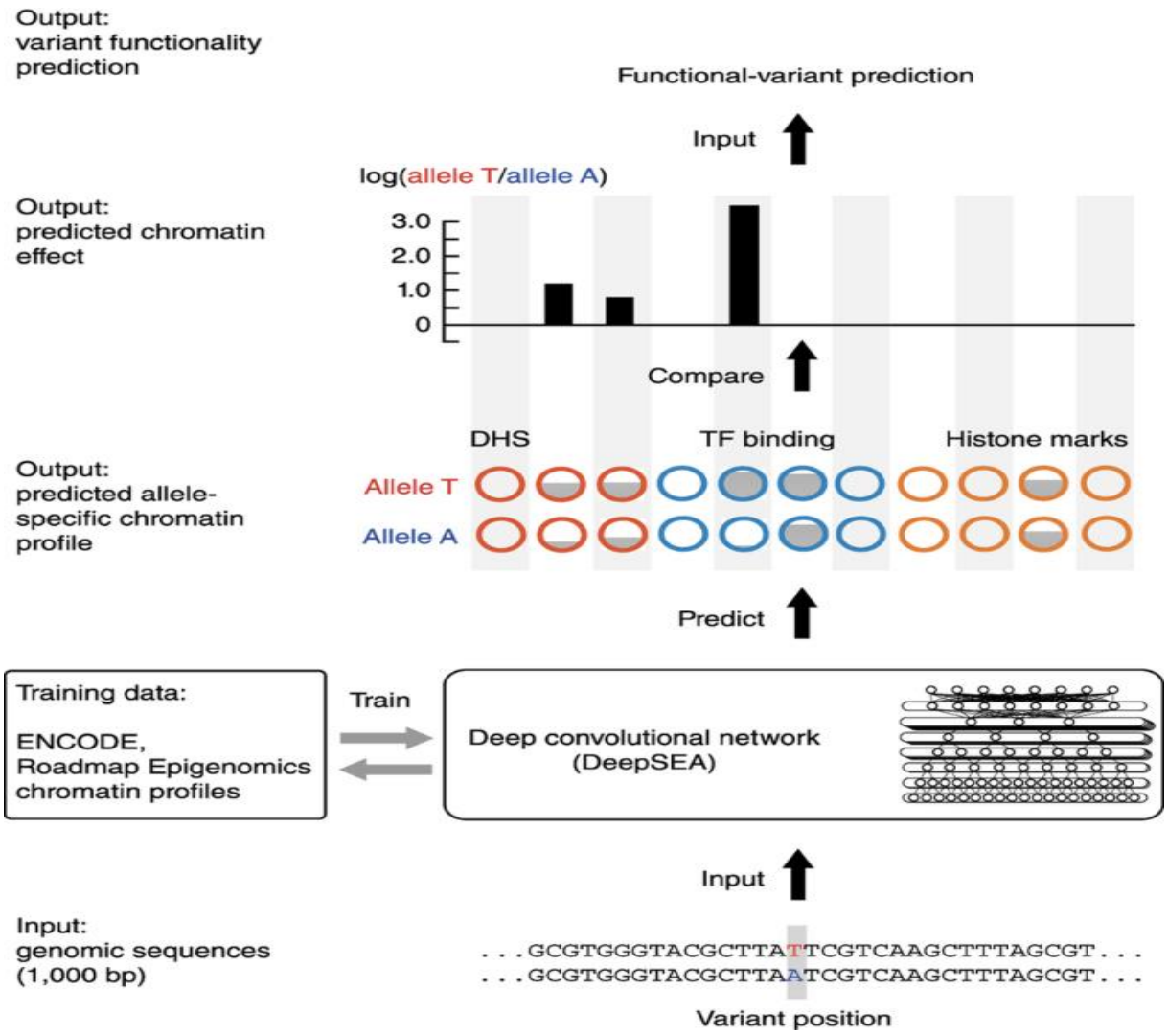


Figure 1.3: Schematic overview of the DeepSEA pipeline, a strategy for predicting chromatin effects of noncoding variants.

Before analyzing the figure 1.3 it's important to understand some concepts.

- **Transcription Factor (TF)** are proteins that binds on a gene regulatory regions to control the transcription of DNA to mRNA. [10]
- **Histone** is protein that wraps around the DNA and helps suit the genome into a cell nucleus. This protein wrap including DNA is called **chromatin**. [11]
- **DNase I hypersensitive sites (DHSs)** are chromatin-sensitive regions by enzyme DNase I, which have lost their concentrated structure, exposed and available DNA in these unique genome areas. This increases the availability of DNA for enzyme degradation such as DNase I. [12]

Explanation of Figure 1.3: The DeepSEA model is for estimating noncoding-variant effects on chromatin. Is pre-trained with a compendium of genome-wide chromatin profiles from the Encyclopedia of DNA Elements (ENCODE) and Roadmap Epigenomics projects (including 690 TF binding profiles for 160 different TFs, 125 DHS profiles and 104 histone-mark profiles).

The model is designed to use as input contexts of genomic sequences with integrated the variant position which is necessary to find the functional effect of non-coding variants. After training the context the model will predict an allele candidate chromatin profile. Where in next phase will help us to detect the chromatin effect of the variant position and also predict the variant functionality.

The table below summarizes the models, the differences, the purpose between them and their accuracy. Table is based on publication [2].

Name	Publication	NN Model	Data	Purpose	Accuracy
DeepFIGV	[7], [2]	CNN	Whole genome sequence	Predict quantitative epigenetic variation	z-scores DNase rho = 0.0802, P = 5.32e–16
DeepVariant	[8], [2]	CNN	Variant caller	Produced more accurate results with greater consistency across a variety of quality metrics	99,45% F1
DeepSEA	[9], [2]	CNN	Noncoding variants (Allelically imbalanced SNPs)	Predicting chromatin effects of noncoding variants.	>95%
Word2Vec	[4], [13]	Skip-Gramm or CBOW, NLP, DL	Corpus	Create a word embedding for words. Find word relations by query.	Necessary Semantic-Syntactic Word Relationship Analysis
Var2Vec (Thesis Model)	[4], [13]	CBOW, NLP, DL	Variants	Create a word embedding for variants. Predict impact variants by query.	Necessary Semantic Variants Relationship Analysis

Table 1.4: Differences between Deep Learning Models.

1.2 Our Solution

An effective way to help researchers extract new knowledge for multiple diseases is to find and analyze these mutations by their properties. Our research question is if the deep learning technology, that has proven to provide impressive results in other domains, can be used for the classification of mutations and prediction of diseases. The rationale behind this effort comes

from the text mining area and using the same principles as document classification we explore the possibility to use the variants of a person in order to classify/identify possible diseases.

In our proposal we create a deep learning model to learn about differences and create a graph of grouped variants. For start we use linear algebra and neural networks to find and calculate the SNP's and visualize the results in a 2D or 3D plot. Specifically, the methodology used is word-embedding which is the formation of weights after training a Natural Language Processing neural network type Continuous Bags of Words for the mutations of homo sapiens.

Before training, the network requires some data as input that in our case is generated from the Ensembl Variant Effect Predictor (VEP). VEP takes the human genetic code (DNA) as input, processes it and at its output mutations are recorded. Then, after training this neural network using the variants, the embedding (weights) were visualized using the Principal Component Analysis (PCA) in two dimensions and in three dimensions. Finally, a formula used to find cosine similarity (Cosine Similarity) in the vectors to detect the similar mutations, in particular the closest ones that have some correlation with each other, new conclusions can be drawn which are from which mutations some of them are affected. In order to find new SNPs that affect, for example, a disease.

2 Background

In section 2 we introduce the methodologies that we used for our solution such as word2Vec, VEP, PCA etc. and provide a short introduction in the corresponding research areas such as biology and artificial intelligence.

2.1 Neural Networks

2.1.1 About neurons

The human cerebrum compromises of around a hundred billion neurons along with a much greater quantity of neuroglia for protecting and assisting the neurons. The network is made up such that a single neuron can be attached to 10,000 more neurons while the signals passed between them take up around 1,000 trillion of synapses. In relation to the sensory system, a synapse is a framework which permits the communication between neurons through signals or for targeting the effector cell by chemical or electrical means. The cell body, dendrite, and an axon make up each mammalian neuron as visible in the following figure. [14]

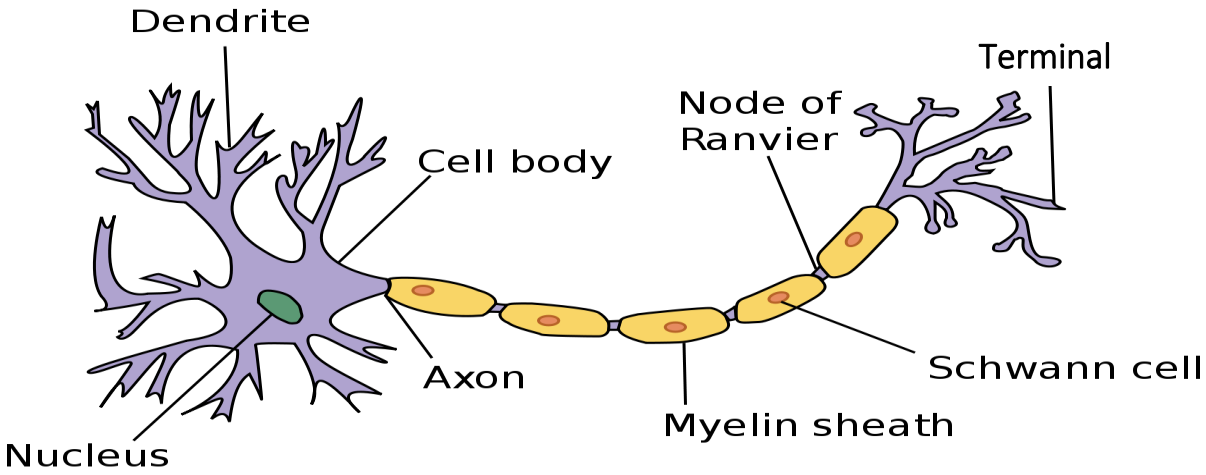


Figure 2.1: Structure of a neuron.

2.1.2 A brief introduction to Artificial Neural Networks

A neural network can be conceptualized as a simulation of interconnected neurons such as a brain. The artificial neural network consists of many interconnected levels of interconnected neurons where they can be modulated during training. The neurons are arranged in layers. In a neural network, there are typically three layers: an input layer. One or more hidden layers (Deep Learning if more than two hidden layers) and an output layer, with a neuron or neurons representing the prediction goals.

The neurons connected with different strengths (or weights) of the connection, it used to answer the question “What pattern or edge in the neuron.” The network learns by analyzing individual records, creating a prediction for each record, and changing the weights whenever

it makes a wrong prediction. The process repeated several times until one or more of the stopping conditions have been met, the network continues to improve its predictions. [15]

Input (X1,X2..Xn)
Weights or Interconnections (W1,W2,..Wn)
Bias (B)
Node - Neuron (Activation Function – f(..))
Output (Y)

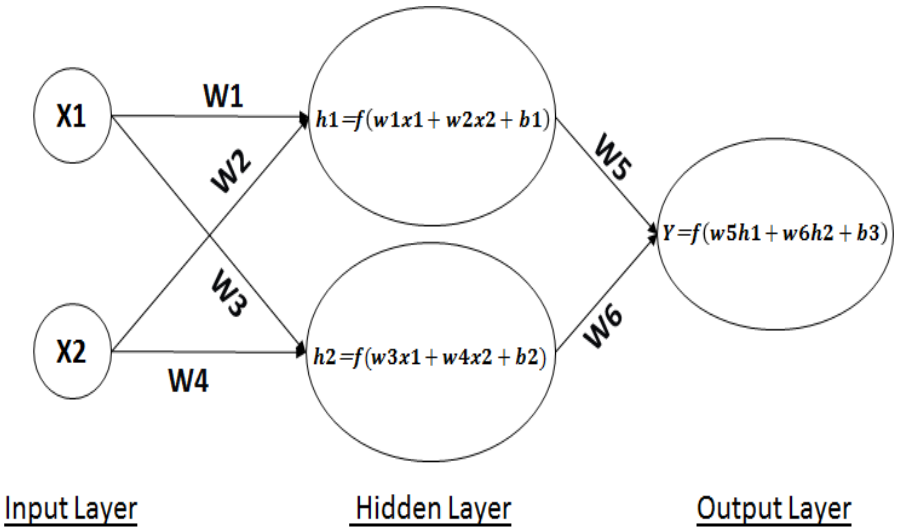


Figure 2.2: Structure of a Neural Network.

2.1.3 Activation Function

Activation function (f) is a function that takes some numbers as input (like input, weights, and bias) a calculates a binary number usually from 0 – 1. If the number is bigger from the threshold (e.g., 0.5), the neuron is activated. One of the most used activation functions are Sigmoid and ReLU.

$$f(x) = \frac{1}{1+e^{-x}} = \frac{e^{x}}{e^{x}+1}$$

The formula of Sigmoid Function

$$f'(x) = f(x) \cdot (1 - f(x))$$

Derivative of sigmoid

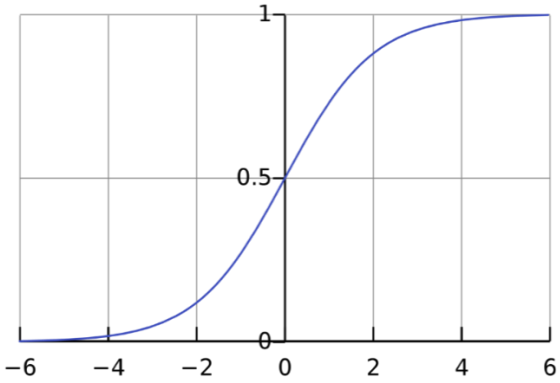


Figure 2.3: Sigmoid Function

[16, p. 425]

2.1.4 Bias

There is one more significant attribute for a neuron. It plays a very special role in the activation of the neuron and goes to the next layer. It is called Bias. The bias represents some attributes of data such as race, gender and color. Which is a factor in activation of a specific neuron. [6][17]

2.1.5 Loss

We first need a way to quantify how well works our neural network to attempt improve it before we train our network. That is the loss. One formula for doing that is Mean Squared Error (MSE).

$$L = MSE = \frac{1}{2}\sum_{i=1}^{n}(YTRUE - Y)^2 = (1 - Y)^2$$

Where **n** is a number of examples, **YTRUE** is the true value of a variable (the correct classified of example, e.g., 1), and **Y** is the prediction variable (output of network). We needed to make better predictions as possible. To do that, we need to train the network to have the best lower loss.

2.1.6 Backpropagation

We want to modify w1. How much loss (L) will change after modification of w1? That we can find it by calculating the partial derivative of $\frac{dL}{dW1}$

$$\frac{dz}{dx} = \frac{dz}{dy} \cdot \frac{dy}{dx}$$

Example: Chain Rule (Leibniz, 1646-1716)

$$\frac{dL}{dW1} = \frac{dL}{dY} \cdot \frac{dY}{dW1}$$

$$\frac{dL}{dY} = \frac{d(1 - Y)^2}{dY} = -2(1 - Y)$$

We remember $Y = f(w5h1 + w2h2 + b3)$. So we cannot solve $\frac{dY}{dW1}$ because w1 affects only h1, so the new chain is:

$$\frac{dY}{dW1} = \frac{dY}{dh1} \cdot \frac{dh1}{dW1}$$

$$\frac{dY}{dh1} = w5 \cdot f'(w5h1 + w6h2 + b3)$$

$$\frac{dh1}{dw1} = x1 \cdot f'(w1x1 + w2x2 + b1)$$

So the final chain will be:

$$\frac{dL}{dw1} = \frac{dL}{dY} \cdot \frac{dY}{dh1} \cdot \frac{dh1}{dw1}$$

[18]

2.1.7 Stochastic Gradient Descent (SGD)

Certain modifications of the weights and biases in order to decrease the loss takes place by the utilization of an optimization algorithm known as Stochastic Gradient Descent (SGD).

$$w1 = w1 - \eta \frac{dL}{dw1}$$

The speed at which we train is controlled by a constant η known as Learning Rate.

- If $\frac{dL}{dW1} > 0$, Loss will be minimized as w1 will decrease
- If $\frac{dL}{dW1} < 0$, Loss will be minimized as w1 will decrease

This needs to be done in case of each weight and bias for enhancing the network forecasts.[18]

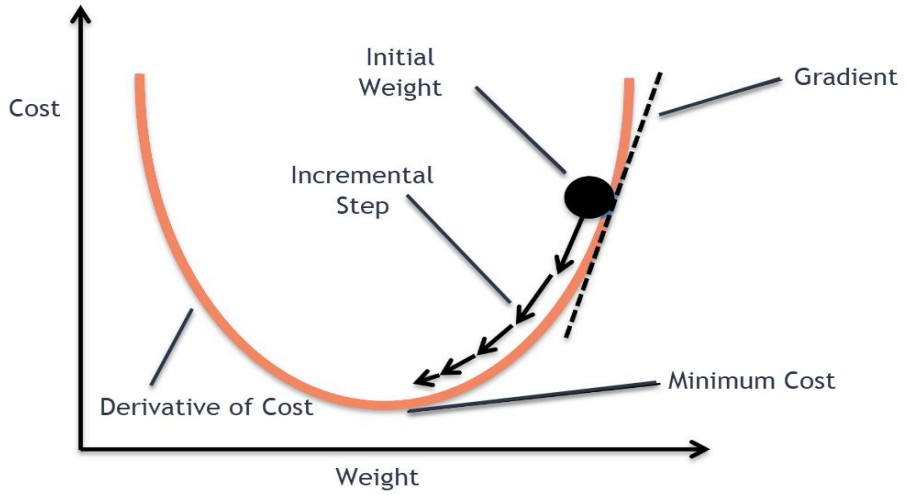


Figure 2.4: Stochastic Gradient Descent (SGD)

2.1.8 Softmax function

SoftMax is a function that turns real numbers into probabilities it used on the output layer. And defined as:

$$S(xi) = \frac{e^{xi}}{\sum_{j=1}^{n} e^{x_j}}$$

2.2 Text Mining and Natural Language Processing (NLP): The Deep Learning approach

Text mining is the research area of extracting interesting and relevant patterns from text sources. Text mining focuses on data processing, data analysis, machine learning, statistics and linguistic computation [19] and it deals with text that is stored in semi-structured and unstructured format in a natural language. In business, academia, web applications, the internet and other areas, text mining techniques are continually applied [20]. For opinion mining, feature extraction, sentiment, predictive, and trend analysis, application areas such as search engines, customer relationship management system, filter emails, product recommendation analysis, spam detection, and social media analytics use text mining. [21]

Natural Language Processing (NLP) is an area of research that tries to give the computers the ability to understands and control speech or text from a natural language. NLP researchers seek to collect information about the use of languages from people to help developers create necessary tools and techniques for computer systems to understand and control natural languages. NLP implementations include a variety of research areas such as computer translation, natural language text encoding and interpretation, user interfaces, CLIR retrieval, language recognition, artificial intelligence, and expert systems (CRIRs). [22] [23]

2.2.1 Word2Vec

The meaning of words in a document is interpreted and vectorized by Word2vec, which implies words of identical meanings have very near distances within a specific context. In the next chapter, we will see the CBOW and Skip-gram model architectures, learning algorithms from word2vec suggested by Mikolov. [24] [25]

2.2.1.1 Word2Vec Models

Numerous NLP functions, such as speech-based tags and machine translation, have shown that word representation from the neural language models has been improved. Such low-dimensional representations are learned in a language model as parameters and trained to optimize the probability of a large raw text corpus. These are then implemented as features alongside hand-crafted features, or used to initialize neural network parameters that target tasks for which significantly less training data is available.

Models mentioned in the Word2Vec by (Mikolov et al [26]), particularly in “skip gram” and “continuous bag of words” (CBOW) models are implemented. However, both models are not word order sensitive. Word embedding models were developed with these models, thus capturing semantic knowledge between words, and pre-training use of these models was seen in several different tasks to lead to substantial improvements. Word2Vec is still a common alternative because of its efficiency and simplicity among other similar approaches like Glove and fastText. The CBOW model takes the mean of the vectors of the input context words and calculates the hidden layer. The model Skip-Gramm was the opposite of the model CBOW, because the target word was in the input layer, while the context words are on the output layer.

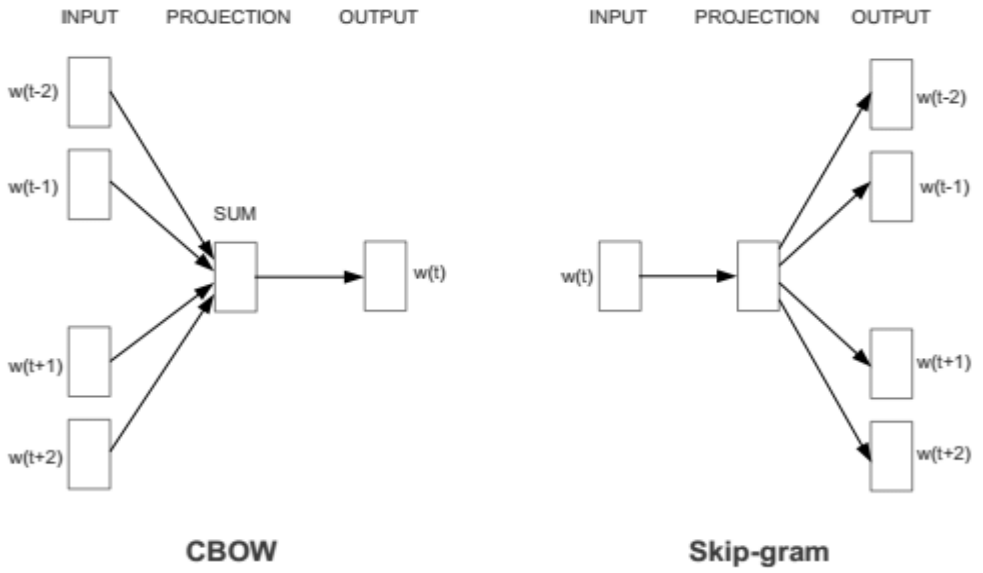


Figure 2.5: CBOW and Skip-Gram Models.

However, because these models are not sensitive to word order, they are not appropriate for tasks like grammar, such as speaking tagging or addiction parsing, since they are designed using these models. This is because "what words go where" is determined by syntax. While semantics than "what words go together." Obviously, several syntactic connections between words cannot be properly captured in a model in which word order is discarded. For example, although the majority of words occur with the word, only names occur exactly after words (e.g., the cat).

This is confirmed by empirical evidence that insensitivity to order does indeed lead to low syntactic representations, whereby systems using Word2Vec models pre-trained provide minor improvements while the computationally far more costly using word order information embeddings of Collobert et al. [27] yields much better results. [28]

2.2.2 Word2Vec Skip-Gram Model

For Skip-Gram, the input is the central word in Word2Vec's skip-gram architecture, with the prediction the context words. Think a collection of terms **W**, where **W (*i*)** would be an input as the central word with context words would be **W (i + 2) and W (i + 1).** If window sliding size is 2, then:

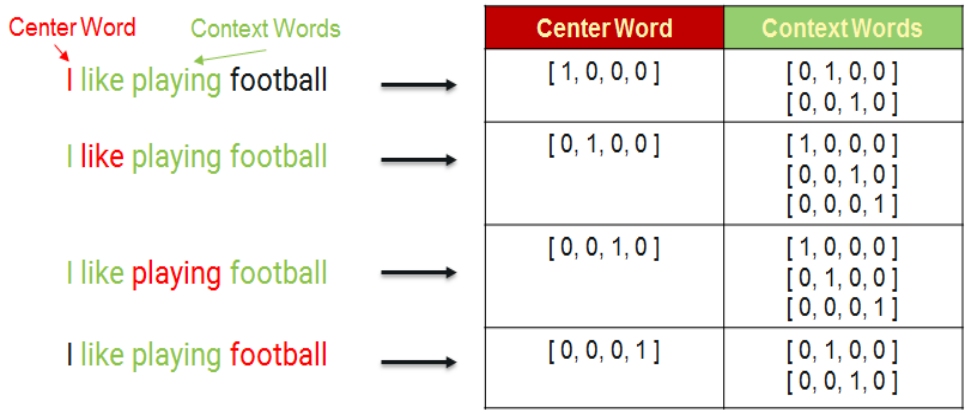


Figure 2.6: Getting context words from a corpus (window slide = 2).

The table below is to understand and remember important parts of the model.

V	Total different words in corpus.
X	Input Layer where input word would be One-Hot encoding.
N	Total neurons present in the secrete layer.
W	Weights among the hidden layers as well as the output layer.
W’	Weight present between the output layer and hidden layer.
y	Output layer with possibilities of almost each word. (SoftMax)

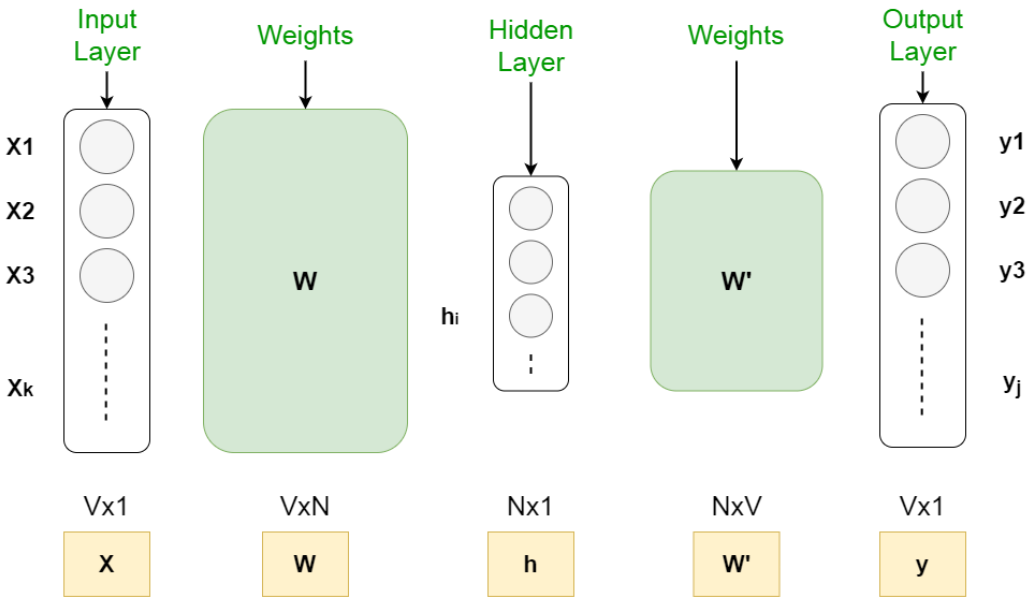


Figure 2.7*: Word2Vec Skip-Gram Model.*

2.2.2.1 Forward Propagation

To get started with forward propagation first we multiply the center word (x) one-hot encoding with basic weight matrix i.e. W to the matrix i.e. h, which is Nx1.

$$h = W^T \cdot X$$

In order to achieve the value of u, the hidden vector "h" is multiplied with the 2nd weight matrix "W".

$$u = W'^{T} \cdot h$$

In order to achieve the value for output layer "y", SoftMax is applied to the layer "u".

- u_j = the j^{th} neuron of "u" layer.
- w_j = the term j^{th} in the vocabulary, indicating "j" as index.
- $V_{wj} = j^{th}$ column of matrix W' i.e. the column next to the W_j
- $u_j = V_{W_{ij}}^{T} \cdot h$

$y = softmax(u)$

$y_i = softmax(u_j)$

y_i stands for the probability of Wj as a context term.

$$P(W_j|W_i) = y_i = \frac{e^{u_j}}{\sum_{j'=1}^{u} e^{u_{j'}}}$$

$P(W_j|W_i)$ indicate the probability of Wj as the context word, with Wi stands for the input word.

Therefore, the main objective is to increase the value for $P(W_{j*}|W_i)$, in which j* stands for the indices of context word.

Goal is to maximize:

$$\prod_{c=1}^{C} \frac{e^{u_{jc*}}}{\sum_{j'=1}^{u} e^{u_{j'}}}$$

Here, the j*c represent the key for unique words that ultimately belongs to the context words. Moreover, these context words begin from 1, 2, 3,…C.

Now, in order to obtain the loss function, we take the –ive log-probability of function that should be minimum.

$$E = -\log\left\{\prod_{c=1}^{C} \frac{e^{u_{jc*}}}{\sum_{j'=1}^{u} e^{u_{j'}}}\right\}$$

E is the loss function. Suppose "t" is the output vector taken for the specific word from the training data. Now, 1 would be located at the context word position, while 0 would be present elsewhere. Here, t_{j*c} would be one of the various context words.

Multiplying $e^{u_{jc*}}$ to t_{jc*}

$$E = -\log\left(\prod_{c=1}^{C} e^{u_{jc*}}\right) + \log\left(\sum_{j'=1}^{u} e^{u_{j'}}\right)^{C}$$

The resultant equation for loss function would be:

$$E = -\sum_{c=1}^{C} u_{jc*} + C \cdot \log\left(\sum_{j'=1}^{u} e^{u_{j'}}\right)$$

2.2.2.2 Back Propagation

The two factors that should be accustomed are "**W**" and the "**W'**". So, the partial derivative should be determined for the loss function in order to apply the method of gradient descent according to the W and W'

Our objective is to determine the $\frac{dE}{dW'}$ and $\frac{dE}{dW}$

$$\frac{dE}{dW'_{ij}} = \frac{dE}{du_j} \cdot \frac{du_j}{dWi_{ij}}$$

$$\frac{dE}{du_j} = -\sum_{c=1}^{C} u_{jc*} + C \cdot \frac{1}{\sum_{j'=1}^{u} e^{u_{j'}}} \cdot \frac{d}{du_j} \sum_{j=1}^{V} e^{u_j}$$

$$\frac{dE}{du_j} = -\sum_{c=1}^{C} 1 + \sum_{j=1}^{V} y_j$$

$$\frac{dE}{du_j} = y_j - t_j = e_j$$

$$\frac{dE}{dW'_{ij}} = e_j \cdot \frac{du_j}{dW'_{ij}} = e_j \cdot \frac{dW'_{ij} * hi}{dW'_{ij}}$$

$$\frac{dE}{dW'_{ij}} = e_j \cdot hi$$

Now finding $\frac{dE}{dW_{ij}}$

$$\frac{dE}{dW_{ij}} = \frac{dE}{du_j} \cdot \frac{du_j}{dW_{ij}}$$

$$\frac{dE}{dW_{ij}} = \frac{dE}{du_j} \cdot \frac{du_j}{dh_i} \cdot \frac{dh_i}{dW_{ij}}$$

$$\frac{dE}{dW_{ij}} = e_j \cdot W'_{ij} \cdot \frac{dW_{ij} \cdot X_i}{dW_{ij}}$$

$$\frac{dE}{dW_{ij}} = e_j \cdot W'_j \cdot X_i$$

[22] [28] [29]

2.2.3 CBOW Model

In Continuous Bag of Word (CBOW) model of Word2Vec, context words are input with prediction as the center word. The text mining process will be the same as skip-gram described

in the figure above 2.6. The difference on this model is we think a set of words W, suppose input are **W (i + 2)** and **W (i + 1)** as conditional words, with center word is **W (i)**. If 2 is the size for sliding window.

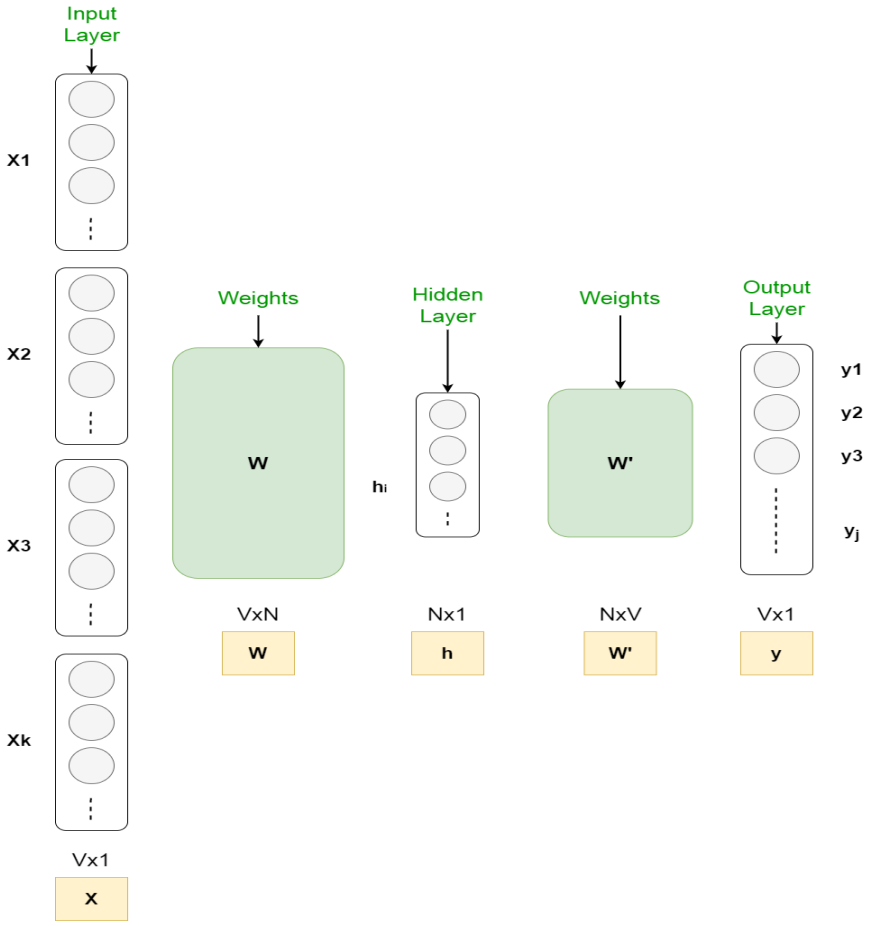


Figure 2.8: Word2Vec CBOW Model.

The table below is to understand and remember important parts of the model.

V	Total different words in corpus.
C	Word window size.
X	Input Layer where input word would be One-Hot encoding.
N	Total neurons present in the secrete layer.
W	Weights among the hidden layers as well as the output layer.
W’	Weight present between the output layer and hidden layer.
y	Output layer with possibilities of almost each word. (Softmax)

2.2.3.1 Forward Propagation

To get started with forwarding propagation of CBOW first, we are multiplying one-hot encoding of every context words (Xi) with the weight matrix W. For every result summed we, divide it by C. Where, the mean for input vectors is weighted by "W" matrix.

$$h = \frac{1}{C} W \cdot \left(\sum_{i=1}^{C} Xi \right)$$

We are now calculating the inputs for every node in the output layer:

$$u_j = {V'_{Wj}}^T \cdot h$$

Where V'_{Wj} indicates the j^{th} column in the $\boldsymbol{W'}$ output matrix.

In the end, the output layer was calculated. The $\boldsymbol{y_i}$ output is obtained by passing the input u_j through soft-max function.

$$y_i = p\left(W_{y_j} | W_1, \dots, W_C\right) = \frac{e^{u_j}}{\sum_{j\prime=1}^{V} e^{u'_j}}$$

After understanding the functioning of forward propagation, the matrices of W and W' can be understood.

2.2.3.2 Back Propagation

In order to understand the weight matrices, first of all the randomly initialized values are learnt, followed by inserting different training examples in the model under keen observation. Then the errors are determined by setting the comparison between real performance and expected performance. This comparison generates the gradient of error, which is measured and then corrected in both of the weight matrixes. This sort of optimization is known as SGD i.e. Stochastic Gradient Descent.

First of all, the loss function is set with the aim of increasing probability of output with the given context of input. So our loss function will be:

$$E = -\log p(W_o | W_I)$$

$$= -u_{j*} - \log \sum_{j'=1}^{V} \exp\left(u_{j\prime}\right)$$

$$= -{V_{Wo}}^T \cdot h - \log \sum_{j'=1}^{V} \exp\left({V_{W_{j'}}}^T \cdot h\right)$$

Where $j*$ is the current output word index, the following step is the derivation of the updated equation for total weight i.e. W', between the hidden layers and output layers. In next we derive W i.e.; the weight intermediates the hidden layer and the input layer.

Update the weight between output layer and hidden layer:

First of all the derivative is calculated for loss function i.e. E of the input to the node j^{th} which belongs at the output layer u_j.

$$\frac{dE}{du_j} = y_j - t_j$$

Where $t_j = 1$ if $j = j^*$ otherwise $t_j = 0$. This simplifies the output layer's prediction error for the "j" node, followed by taking out the derivative of E for the $\boldsymbol{W'_{ij}}$ output weight by following the chain rule.

$$\frac{dE}{dW'_{ij}} = \frac{dE}{du_j} \cdot \frac{du_j}{dW'_{ij}}$$

$$= (y_j - t_j) \cdot h_i$$

At this point, the gradient of $\boldsymbol{W'_{ij}}$ the weight of arbitrary output is obtained and now the stochastic gradient can be defined for the descent equation.

$$W'_{ij} = W'_{ij} - \eta \cdot (y_j - t_j) \cdot h_i$$

or

$$V'_{W_j} = V'_{W_j} - \eta \cdot (y_j - t_j) \cdot h$$

Here, the η > 0 stands for the learning rate, h_i stands for the hidden layer's i^{th}column, and the V'_{W_j}indicates the w_j' output vector.

Keeping it under consideration, this updated equation suggests reviewing each word of vocabulary and analyzes their output probability i.e.y_j, followed by comparing y_j with t_j that is its expected output as 0 or 1.

- In case, $y_j > t_j$ ("overestimating"), a portion of hidden vector h is subtracted from the V'_{w_j}, leading to make the V'_{w_j} away from V_{w_I}.
- In case, $y_j < t_j$ ("underestimating", as it can be accurate if $t_j = 1$ and $W_j = W_O$, followed by addition of "h" to V'_{W_O}, leading to make the V'_{W_O} closer to V_{W_I}.
- In case the y_j stands closer to t_j, there will be just a slight modification in weights as per the updated equation.

Must remember that the input vector (V_W) is different from the output vector i.e. V'_W

Updating the input-hidden layer weights

Here, the major focus is laid on derivation of updated equation for W_{ij}, which is the input weight. First of all the derivative of E is calculated according the arbitrary hidden nodes i.e. h_i by following the chain rule.

$$\frac{dE}{dh_i} = \sum_{j=1}^{V} \frac{dE}{du_j} \cdot \frac{du_j}{dh_i}$$

$$= \sum_{j=1}^{V} (y_j - t_j) \cdot W'_{ij}$$

Here, the calculation is completed by following the fact that nodeh_i of hidden layer is attached with every node f output layer, hence every prediction error needs to be integrated. Finally the derivative of E is calculated accordingW_{k_i}, which is arbitrary input weight.

$$\frac{dE}{dW_{k_i}} = \frac{dE}{dh_i} \cdot \frac{dh_i}{dW_{k_i}}$$

$$= \sum_{j=1}^{V} (y_j - t_j) \cdot W'_{ij} \cdot \frac{1}{C} \cdot X_k$$

$$= \frac{1}{C} (x \cdot EH)$$

Where EH stands for the elements' N-Dimensional vector $\sum_{j=1}^{V} (y_j - t_j) \cdot W'_{ij} \cdot \frac{1}{C} \cdot X_k$ from $i = 1, \dots, N$. Although the X inputs are one-hot encoded and there will be just one non-zero row i.e. $N \times V$ matrix $\frac{1}{C} (x \cdot EH)$. The final SGD equation for the input weight is

$$V'_{W_{I,c}} = V'_{W_{I,c}} - \eta \cdot \frac{1}{C} \cdot EH$$

Where $W_{I,c}$ is the c^{th} word in the input context. [26]

The CBOW model was used in our problem. The reason for the selection of CBOW is that the basic information is the description of the variants (contexts) and the purpose is to generate vectors for each SNP (e.g. rsXXXX). In Word Embedding templates before training the model needs to be pre-processed. More specifically, the text has to be broken into words and after the number of the window defined by an expert, lists are formed - context with each central word. The difference in our case is that since we have the information of the variants, the most important fields were selected based on their correlation and the experts. Then we assign each word to a number. These numbers are the context of the neural network where it corresponds to an SNP target [4] [24].

2.3 Principal Component Analysis (PCA)

Principal components analysis is a method used for transforming a number of possibly correlated variables in a small number of variables known as main component using mathematical principles. The original dataset, which may have involved many variables, can often be interpreted in just a few variables, using mathematical projection. The basic concept of the main component analysis is to minimize the dimensionality of a dataset, in which several similar variables occur, while preserving as much variance as possible in the data set. This reduction comes by converting the uncorrelated and ordered main components into a new collection of variables which retained from some first component that contains the most of variation.

The main use of Principal Component Analysis (PCA) is to reduce the dimensions of a dataset that includes a huge number of associated variables and keeping best possible variance.

The goals of PCA are to:

1. Extract the most important information from the dataset.
2. Compress the size of the dataset by keeping only most important data.
3. Simplify the description of the dataset.
4. Inspect the structure of the observations and the variables.
5. Compress the data by decreasing the number of dimensions, without losing much information.
6. This technique used in image compression.

We have to be through in statistics and matrix algebra in order to analyze the data by the Principal Component Analysis. In neural networks an n-dimensional vector with numeric characteristics represents an object or context. The incorporating semantic similarities between texts holds an important position. [30]

2.4 Cosine similarity

One of the most common utilized standards for information fetching is Cosine similarity. A text file is transformed in a vector of terminologies by this standard. Through the utilization of this standard, the resemblance of two files can be determined by computing the cosine value among the term vectors of both the files. This standard can be performed on any two blocks of text (whole document, paragraph, or single line). In instance of a search engine, the likeness factor of the input query and the documents are retrieved from most resembling to lower resemblance. The greater the resemblance score among the document's and query's term vectors, the greater the resemblance in both the said excerpts.

The word or term searched should be accommodated while utilizing cosine similarity method for determining the resemblance ratio between document and the search term given by the user. The semantic meaning of the search term might not be perfectly accurate using cosine similarity. The execution of cosine similarity standard may at times give undependable outcomes between two term vectors in terms of syntax. Comparing the syntax might not provide solution for the problem of semantic meaning. For additional procedure, i.e., data fetching framework, it might deliver false outcome and cause corrupting in its presentation.

According to document-query scenarios, the document is portrayed in the form of a term vector where the proportions of the vector correspond to the terms present in the document. The value of the proportion is associated to the frequency of term in the document. The representation of the document in the form of a vector is:

$$\vec{d} = (W_{d0}, W_{d1}, \dots, W_{dk})$$

In a similar way as per the document, the representation of query is given as follows:

$$\vec{q} = (W_{q0}, W_{q1}, \dots, W_{qk})$$

the occurrence of terms inside a document is given by w_{di} and w_{qi} $(0 \leq i \leq k)$ which are float numbers, whereas the value of the proportion is associated to the frequency of term in the

document. On the basis of vector similarity, the resemblance of two vectors is determined by: [31]

$$similarity(\vec{q}, \vec{d}) = \cos(\theta) = \frac{\vec{q} \cdot \vec{d}}{|\vec{q}| \cdot |\vec{d}|} = \frac{\sum_{k=1}^{t} w_{qk} \times w_{dk}}{\sqrt{\sum_{k=1}^{t} (w_{qk})^2} \cdot \sqrt{\sum_{k=1}^{t} (w_{dk})^2}}$$

2.5 Genetics

2.5.1 A Brief Introduction

All in all, genetics is generally viewed as a zone of science that encourages one to comprehend the systems and techniques utilized in characteristics acquired from parents to children it additionally crosses with the innate and hereditary varieties in living life forms. It is emphatically connected to the utilization of genetics to human undertakings. While, chromosomes are an arranged framework comprised in the cell that contains various genes, consisting long chains of single atoms of deoxyribonucleic corrosive (DNA) or, at times, ribonucleic corrosive (RNA) and the connected protein. Many bacterial chromosomes are basically a solitary, round, twofold linked DNA generally appended to the plasma film, which consists various qualities.

Hereditary data for the most part has its own job towards genotype (duplication), phenotype (gene characteristic), and modification. Ordinarily, by correlation, DNA is a long atom, comprising of our unique hereditary code and accumulating hereditary data. This comprises of subunit links known as nucleotides (otherwise called "bases") comprising of a phosphate particle, a sugar atom (5-carbon) and a nitrogen base. Each filament comprises of a long arrangement of the four essential structure squares or 'bases.' DNA is a part of its information accumulated in DNA as a code comprising of four synthetic bases: thymine (T), cytosine (C), guanine (G) and adenine (A).

Technique for the DNA record and RNA interpretation, earlier known as a core dogma of sub-atomic science, will at first go through DNA for the advancement of protein. This core dogma portrays the two stage cycle, record and interpretation through which protein is produced by data in genes: DNA-RNA protein. RNA is a functioning atom with long nucleotide links. It has ribose sugar, phosphate., and nitrogen base. In any case, now and again, there are blunders that cause change when a mistake happened during DNA replication/DNA record. Two major change related circumstances; DNA duplication blunders/DNA record mistake and synthetic or actual impacts brought about by explicit transformation related variables or unconstrained change. In like manner, we may discover different writings while examining current genetics [5].

2.5.2 Human Genome Project

The principle point of the Human Genome Project is to investigate internationally the formation of the human genome and to group all qualities of the human genomes. In 2001 to 2003, Venter and Francis Colleins declared the full draft of the human genome pattern with a

98% human genome sequenced at an accuracy of over 99,9%. In previous occasions, the HGP was exceptionally eager and pointed toward building up a human hereditary planning framework, finding 3 billion human genomic nucleotides and describing the whole collection of these by 24 chromosomes in 2005 [32].

2.5.3 Ensemble Variant Effect Predictor: Annotating Variant Effects

The Ensemble Variant Effect Predictor (VEP) is an apparatus which comments on, assess and organize genomic variations, including non-coding areas. VEP foresees precisely the impacts of grouping variations on records, protein items, administrative areas, and restricting themes by utilizing the Ensemble information bases' top notch, wide degree, and incorporated plan. It additionally considers correlation with a wide arrangement of existing freely available variety information inside Ensemble to give understanding into populace and genealogical hereditary qualities, aggregates and infection. VEP is free and an open source. It is accessible by means of a straightforward web interface (http:/www.ensembl.org/vep), an adaptable downloadable bundle and the Perl and REST application programming (API) administrations of both Ensemble.

VEP clarifies two distinct sorts of genomic variations: (1) arrangement variations with one of a kind and clearly characterized changes (including SNVs, additions, cancellations, various base pair replacements, microsatellites, and tandem redundancy); and (2) bigger primary variations (bigger than fifty nucleotides long) containing contrasts for the quantity of duplicates or inclusions and DNA erasures. The VEP returns specified examination for all information variations for consequences for records, proteins, and administrative areas. Included for distinguished or comparable variations are allele frequencies and data about infection or aggregate.

The VEP discoveries contain a wide scope of information applicable to genes and records (Table 2.10). Record sets can be utilized on an essential reference get together or ALT grouping, yet naturally, the VEP chooses Ensemble explanation.

Property	Description
Gene ID	Ensembl stable identifier for affected gene
Gene symbol	Common name for gene, e.g., from HGNC
Transcript ID	Ensembl stable identifier for affected transcript
RefSeq ID	NCBI RefSeq identifier for affected transcript
CCDS ID	Consensus coding sequence (CCDS) identifier uniting Havana, Ensembl, and NCBI
Biotype	GENCODE biotype of affected transcript
cDNA coordinates	Coordinates of input variant in unprocessed cDNA
CDS coordinates	Coordinates of input variant in processed coding sequence (CDS)
Distance	Distance to transcript if variant falls outside transcript boundaries
Consequence type	SO consequence type of input variant allele on transcript
Exon	Number(s) of affected exon(s)
Intron	Number(s) of affected intron(s)
TSL	Transcript Support Level (TSL) highlights well-supported and poorly supported transcript models
APPRIS	Annotation principle splice isoforms (APPRIS) is a system to annotate alternatively spliced transcripts based on a range of computational methods, assigning primary and alternative statuses to transcripts
HGVS	HGVS notations for input variant relative to the coding sequence
Phenotype	Flag indicating known association with a phenotype or disease

Table 2.6: Gene and transcript-related fields reported by the VEP.

The VEP filters the information bases for the Ensemble Variability, which comprise a large assortment of freely accessible information on germ line and substantial inconstancy in vertebrates. Ensemble fuses variations from dbSNP and different hotspots for 20 life forms, and quality controls. Explicit human examples incorporate COSMIC and Human Gene Mutation Database varieties, and underlying variations, also duplicate number variations from the Genomic Variants Database information base.

The VEP may thusly contrast a great many variations which characterize the ones already referenced. The VEP covers allele frequencies as per the activities 1000 Genomes, NHLBI exome sequencing and ExAC. These can be utilized as channels, taking into account the avoidance of explicit variations as pathogenic applicants. The VEP gives PubMed identifiers to referred variations, which likewise explain those related with an aggregate, confusion, or attribute utilizing information from Orphanet, OMIM, the GWAS Catalog, and other information sources. Clinical condition of importance given by ClinVar is additionally basic for human variations. [1]

Property	Description
Variant ID	External identifier for variant co-located with input, e.g., rsID from dbSNP
Somatic	Somatic status of co-located variant
GMAF	Global minor allele and frequency of co-located variant from combined 1000 Genomes phase 3 populations
Other frequencies	Frequency data from continental level 1000 Genomes phase 3 data and two NHLBI–Exome Sequencing Project populations
Clinical significance	Clinical significance status of co-located variant as reported by ClinVar
Phenotype	Flag indicating known association with a phenotype or disease
PubMed ID	NCBI PubMed IDs of publications citing co-located variant

Table 2.7: Co-located variant-related fields reported by the VEP

3 Methodology

In this chapter, we explain the methodology we followed to figure, identify and categorize the variants and correlations of mutations from a patient like the example above, in order to study them, draw new conclusions and find their similarities as well as how one mutation can affect another.

At the beginning, we should download "00-common_all.vcf.gf" which is a file with the human variations in VCF format without clinical assertions that have been mapped to assemblies GRCh37 and GRCh38, provided by dbSNP [33]. Furthermore, we should extract data from the file so is necessary to create a script that reads this file and selects the fields #CHROM, POS, ID, REF and ALT [34].

Once we have this data, we use it as input to the Ensembl Variant Effect Predictor (VEP). Next, when VEP data processing is finished and the variants was discovered from VEP.

We selected the Continuous Bags of Words (CBOW), deep learning model that takes as an input multi-words contexts (developer-defined list which contains numerical representations of words). When computing the hidden layer output, the CBOW model takes the average of the vectors of the input context words and use the average vector as the output. The main criteria of this model is that the input layer use as basic information the selected attributes of the variants context from VEP database in order to generate vectors for each target SNP (e.g. rsXXXX). In word embedding before training the model we need to preprocess the data/text e.g. the sentences will be broken into words and after the length of window (window=context where words are encoded in unique numbers) defined by an expert or experiments, word lists are formed also and a context with each central word.

The difference in our case is that we have the variants from VEP, the most important fields were selected based on their correlation and the experts. These fields are described in the sections with the Experiments below. Also, we have assigned each word/variant to a number. These numbers are the context of the neural network where it corresponds to an SNP target [4].

Finally, after training the CBOW model and embedding was formed we extract the word-embedding (weights between Input Layer and Hidden Layer) and use those embedding's as input to the Principal Component Analysis (PCA) in order to represent the output components with a scatter plot [30]. Figure 3.1 depicts the high-level architecture of our methodology while the following sub=sections describe the main steps of our implementation.

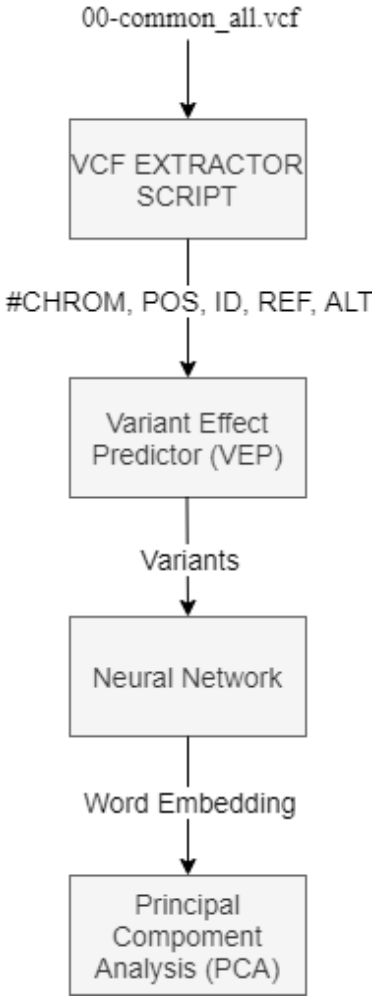


Figure 3.1: Methodology Architecture

3.1 Prepare variation data for VEP

Step 1. Download all variation dataset from ftp://ftp.ncbi.nih.gov/snp/organisms/human_9606/VCF/00-common_all.vcf.gz

Step 2. Extract data in a folder (file "00-common_all.vcf").

Step 3. The following script generates some files in 'data/generated/chrX.vcf'. Those files contain information about variants. Specific 'CHROMOSOME', 'POSITION', 'ID', 'REFERENCE' and 'ALTERNATIVE'. That files will be used as input on VEP.

3.2 Installing VEP

```
1. git clone https://github.com/Ensembl/ensembl-vep.git
2. cd ensembl-vep
3. chmod +x INSTALL.pl
4. perl INSTALL.pl
```

While install you must select those numbers-files:

```
296: homo_sapiens_merged_vep_94_GRCh38.tar.gz
298: homo_sapiens_refseq_vep_94_GRCh38.tar.gz
300: homo_sapiens_vep_94_GRCh38.tar.gz
```

If everything is completed download GRCh38 fasta assembly from ensembl and run example:

```
1. mkdir fasta
2. cd fasta
3. curl -O ftp://ftp.ensembl.org/pub/release-
   96/fasta/homo_sapiens/dna/Homo_sapiens.GRCh38.dna.primary_assembly.fa
   .gz
4. bgzip -d Homo_sapiens.GRCh38.dna.primary_assembly.fa.gz
5. cd ../
6. ./vep -i examples/homo_sapiens_GRCh38.vcf --cache --force_overwrite -
   -merged --offline --everything --fasta
   ./fasta/Homo_sapiens.GRCh38.dna.primary_assembly.fa.gz --assembly
   GRCh38 -o stdout
```

Now, we can run VEP for all snps of Homo Sapiens. To do that copy the all generated files "chrX.vcf" from previous steps and paste it on ensembl-vep/input/. Run it and wait to finish some hours/days and you will see all chromosome variants in ensembl-vep /output folder.

3.3 Data Preprocessing

According to Word2Vec CBOW model, the data was used is a text document. So, for start the developer must define a length of the sentence that is necessary to start processing a whole document that he wants to process, a step called "sliding window". The sliding window is a container that contains few words encoded in numbers. For every word in this container, we need to extract the context words and center words. The center word, otherwise called "target word" is the mark (index) number in whole document words. The previous and next words among the center word on the current sliding window are the Context words of the window.

The difference in our methodology is the variants that we have to get from VEP were selected by an expert along with the correlation of attributes. The selected attributes are described in the experiment section. The start idea was the attributes will be used as contexts (encoded as unique numbers) and the SNP as all words in the vocabulary. In shortcut context (selected attributes) targets a SNP (rsXXXX). That information was used as input in our model. But we conclude with the final idea in third experiment, the common attributes among SNP's will be used as SNP indexing. Those indexes are 1-window contexts and the target is one-hot encoding of total SNP's.

3.4 Model Selection

The CBOW model usually is taken as input multi-words contexts. The model predicts the centre word from its surrounding context [35]. When computing the hidden layer output, the model takes the average of the vectors of the input context words and use the average vector as the output. The CBOW model was used in our problem than Skip-Gramm. The reason this

model was used is that the basic information is the description of the variants (contexts) and the purpose is to generate vectors for each target SNP (e.g. rsXXXX) [13] [5].

3.5 Validation

Various word embedding models generate different vector representations. It is a matter of checking how well the concept of the perceived similarity is taken from the word vector representations and validating the distribution hypothesis, where the meaning of the terms is linked with the context in which they occur. For this latter, it is still unclear how similarities are simulated through distributive semantic models.

The cosine similarity is as an inspector. To clarify, inspector measures the association between all dimensions of the vector, regardless of their meaning for a given word pair or for a semantic group [35]. So, we pass the word embedding results into cosine similarity where will compare all SNPs vector and will select the most similar for one selected SNP. To validate this, we must compare the SNP attribute for instance REF, ALT and if are almost the same we have a valid result. This process must be done for every SNP in the dataset to have a clear image about them.

4 First Experiment

4.1 Problem definition

On the first experiment, the objective is to cluster all the known variants and to preview all those datasets by their unique attributes. We need a solution to be able to visualize - plot those datasets to help bioinformaticians and biologists to understand more about mutations.

Another challenge after accomplishing the visualization is that we need a way to be able to find if a mutation has some correlation (or the opposite) with all other mutations. If they are similar by their attributes we possibly can watch and study them to found new conclusions about them (e.g., a possible cancer SNP).

According to the architecture we introduce in the previous chapter we believe a deep learning neural model and more specific a natural language processing deep neural model will be able to help us analyses our data while the PCA to plot this type of data. Such a model is designed to process documents and produce vectors of words to be able to plot them to find the similarities between them. In this experiment, we will try to use variant data instead of words to be able to produce vectors of variants.

4.2 Data

The table below shows the results of variants data generated from VEP. Some of these fields will be use in our neural network as input. Table is available on VEP website [34].

Columns	Description	Used
LOCATION (or CHROMOSOME)	Location of variant in standard coordinate format (chr:start or chr:start-end)	YES
Allele	The variant allele used to calculate the consequence.	YES
Gene	Stable ID of affected gene.	YES
Feature	Stable ID of feature.	YES
Consequence	Consequence type.	YES
AF	Frequency of existing variant in 1000 Genomes combined population.	YES
GIVEN_REF	Reference allele from input.	YES
SOURCE (Ensembl)	Source of transcript.	YES
IMPACT	Subjective impact classification of consequence type.	YES
STRAND	Strand of the feature (1/-1).	YES
VARIANT_CLASS	SO variant class.	YES
SYMBOL	Gene symbol (e.g. HGNC).	YES
SYMBOL_SOURCE	Source of gene symbol	NO
BIOTYPE	Biotype of transcript or regulatory feature.	YES
CANONICAL	Indicates if transcript is canonical for this gene.	YES
TSL	Transcript support level.	YES
BIOTYPE	Biotype of transcript or regulatory feature.	YES
USED_REF	Reference allele as used to get consequences.	YES
INTRON	Intron number(s) / total.	YES

EXON	Exon number(s) / total.	YES
CLIN_SIG	ClinVar clinical significance of the dbSNP variant.	YES
SOMATIC	Somatic status of existing variant.	NO
GENE_PHENO	Indicates if gene is associated with a phenotype, disease or trait.	YES
MOTIF_NAME	The source and identifier of a transcription factor binding profile (TFBP) aligned at this position.	NO
MOTIF_POS	The relative position of the variation in the aligned TFBP.	NO
HIGH_INF_POS	A flag indicating if the variant falls in a high information position of the TFBP.	NO
BAM_EDIT	Indicates success or failure of edit using BAM file.	NO
miRNA	SO terms of overlapped miRNA secondary structure feature(s).	NO

Table 4.1: Used fields and their descriptions from VEP.

4.3 Scripts

```
1. from tensorflow.keras import utils
2.
3. def generate_data():
4.     total_rs = len(rs)
5.
6.     for counter in range(0, total_rs):
7.         x = np.array(  [ wids[counter] ] )
8.         y = np.array( [ utils.to_categorical(counter, total_rs) ]  )
9.
10.          yield x, y
11.
12.
13.  # build CBOW architecture
14.  cbow = tf.keras.models.Sequential([
15.    tf.keras.layers.Embedding(input_dim=vocab_size,
   output_dim=embed_size, input_length=window_size ),
16.    tf.keras.layers.Lambda(lambda x: tf.keras.backend.mean(x,
   axis=1), output_shape=(embed_size,)),
17.    tf.keras.layers.Dense(vocab_size, activation='softmax')
18.  ])
19.
20.
21.  cbow.compile(optimizer='rmsprop',
22.                loss='categorical_crossentropy')
23.
24.  def training():
25.      for epoch in range(1, 150):
26.          loss = 0.
27.          i = 0
28.          for x, y in generate_data():
29.              i += 1
30.              loss += cbow.train_on_batch(x, y)
31.              if i % 100000 == 0:
32.                  print('Processed {} (context, variants)
   pairs'.format(i))
33.
34.          print('Epoch:', epoch, '\tLoss:', loss)
35.  print()
```

```
36.
37.  training()
38.
39.  weights = cbow.get_weights()[0] #Word-Embedding
```

Code 4.2: A python implementation of CBOW deep learning model

In the code section above, we have reproduced an example of a CBOW deep learning model. In the function generate_data(), the SNP's encodes to one-hot array (zeros and ones) by its index on whole document. Nextly, we create the model architecture were defined by the vocabulary length, contect length, embedding size and loss function. On training() function, it is necessary to define the epochs to start training. At the end, when training have finished we extract word embedding from our deep learning model.

4.4 Results

In the first experiment for chromosome 22 using a Continues Bag Of Words Model after 5 epochs, and using an NVIDIA CUDA 2080ti card and 64GB RAM over a period of four weeks shows that our model succeeds a small variation description (~4%) and the results were not qualitative. Due to the long time required for model training and processing of huge volumes of data, it resulted in invalid results. To be able to assume that the results are valid would require more time and more experiments for the reliability of the results. To visualize the accuracy of the results we use Cosine Similarity and compare our samples (i.e., SNPs) with a database e.g., of NCBI we then compare them in terms of Alleles, Position and Variation Type characteristics. If most SNP's are close by and there is a lot of similarity between them, then our results are quite accurate.

After training variation data of chromosome 22 we get 2 files (vectors22.tsv and rs22.dat). We use them as input to PCA, or you can use also TensorFlow Projector to get the 2-Dimension or 3-Dimension of chromosome 22 embedding.

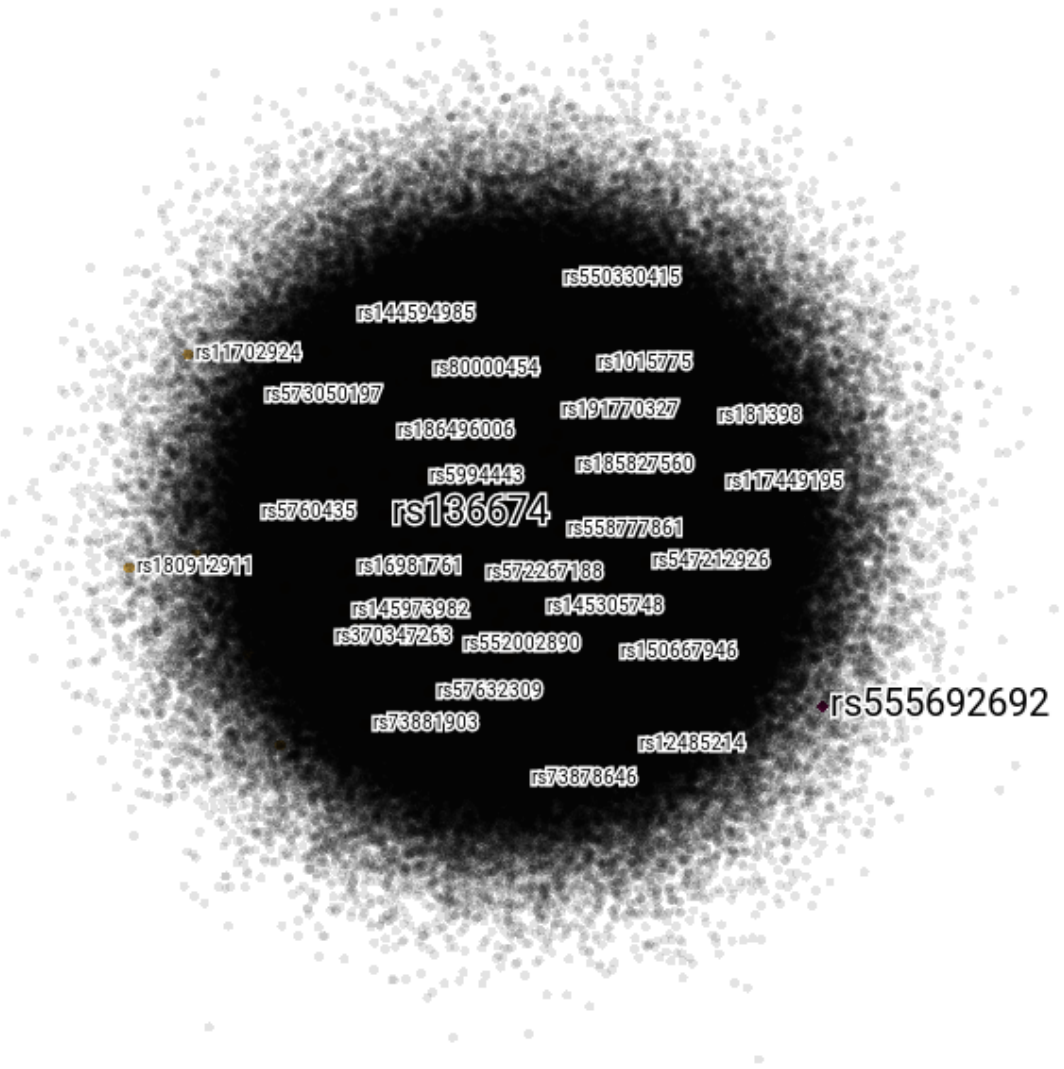


Figure 4.3: Embedding Projection of Chromosome 22 variants.

In the figure above, it depicts the embedding results after training our model with variation data of chromosome 22. In fact, we choose a random SNP and we search and watch the closest SNP's using Cosine Similarity. The results are not too accurate concluding that the training was not enough for a better view. We believe that we need more time for the training since the variation data was too big. The main problem was that the specific model in Tensorflow does not work with too large data for all chromosomes. The limiting factor is the size of the final matrix in a node since its size is calculated by this formula {number of words} *{dimensions}*{float size}. In this experiment, we used 21 attributes as context, 1083881 number of words, with 101 dimensions and only 5 epochs, a setup that forced our infrastructure to train the model for four weeks. With our estimates according to the 2nd experiment, 120 epochs are a good starting point which needs about 2 years with the current hardware to provide result. Also training it cannot be going faster cause the limit factor {number of words} *{dimensions}*{float size}.

SNP's	CLINVAR
rs136674	chr22:g.45237239C>T
rs555692692	chr22:g.20013752C>G
rs11702924	chr22:g.45929200T>C
rs144594985	chr22:g.24506083A>G
rs550330415	chr22:g.37486227C>T
rs573050197	chr22:g.21187073A>C
rs80000454	chr22:g.42176511C>T
rs1015775	chr22:g.37251477A>G chr22:g.37251477A>T
rs186496006	chr22:g.19835053T>C
rs191770327	chr22:g.47551418G>A
rs181398	chr22:g.18224781C>T
rs5760435	chr22:g.24858500C>A chr22:g.24858500C>G chr22:g.24858500C>T
rs5994443	chr22:g.32275353T>C
rs185827560	chr22:g.20043768C>G chr22:g.20043768C>T
rs117449195	chr22:g.47381883C>T
rs558777861	chr22:g.24491342G>T
rs16981761	chr22:g.26725869C>T
rs572267188	chr22:g.45014431G>T
rs547212926	chr22:g.30188731G>A
rs145973982	chr22:g.32187810G>T chr22:g.32187810G>A
rs145305748	chr22:g.41187810C>A chr22:g.41187810C>T
rs370347263	chr22:g.50696649G>A
rs552002890	chr22:g.34104172G>A
rs150667946	chr22:g.43438229G>A
rs57632309	chr22:g.50579775_50579776del chr22:g.50579776dup chr22:g.50579773_50579776del chr22:g.50579776del chr22:g.50579774_50579776del
rs73881903	chr22:g.33269832G>A
rs12485214	chr22:g.48957989G>T chr22:g.48957989G>A
rs73878646	chr22:g.23442691G>T
rs180912911	chr22:g.47177568C>A

Table 4.4: Getting and compare variants from a web database.

In table above we use the myVariants API in Python 3. We select to download from ClinVar database. Now we analyze and compare similarities from the table data. "chrXX" is the Chromosome and "X>Y" is allele. According the top first table rows the alleles are not the same so results are not valid.

5 Second Experiment

5.1 Problem definition

Some single nucleotide polymorphism (SNP) are very similar so we need to use some mathematical methodologies like “Independent Samples T” and “p-value”, to be able to verify if are really similar [36] but we are not sure for the results because in genetics we must find if a SNP affects the protein activity [37] of a target disease.

On the second experiment, the objective is to categorize a SNP dataset to a specific disease. According to dataset where is selected by the criteria if exists a cancerous SNP. An expert in the field of bioinformatics provided to us a list of mutations that fall into one of the three following categories LIKELY_BENIGN: 11119, BENIGN: 3547 and PATHOGENIC: 14499 - TOTAL: 29165 SNP’s. We discovered after using Principal Component Analysis (PCA) and we plot it, that it’s hard to cluster only “cancer” and “possibly cancer” variants and preview all those datasets by their unique attributes.

One another challenge after the plotting those cancerous SNPs we need to verify the results and somehow be able to find if a mutation has some correlation (or the opposite) with another and if they are similar so that we can study them and possibly find new conclusions about these (e.g. a patient with mutation rs1234 has a predisposition to cancer).

According to the architecture we introduce in the methodology chapter we believe a deep learning neural model and more specific a natural language processing deep neural model will be able to help us analyses our data while the PCA to plot this type of data. Such a model is designed to process documents and produce vectors of words to be able to plot them to find the similarities between them. Again in the second experiment, we will try to use variant data instead of words to be able to produce vectors of variants.

5.2 Data

Some of those fields (specific RS and Classification) will be used in the neural network as input. The dataset fields are described below:

VariantType
Gene
RS
Chr
Start
End
Ref_Allele
Alt_Allele

Submisson_Review
Num_Submitters
Cancer_MedGen_Pheno
Classification

Table 5.1: Cancer Variants

5.3 Results

In the second experiment, we have a database of cancerous SNPs and possibly cancerous. This data has many similarities in their information and consist of a multidimensional structure. To begin with, it was analyzed by PCA, resulting in the observation that the sample is quite homogeneous in the first phase. Then, contexts and labels where created for these groups ("Cancer Variants" and "Likely Benign Cancer Variants" group). The need to separate our dataset in one-hot (binary encode) groups are the cause of the CBOW model architecture that requires a target context to start processing it. The context and labels where that are the input for the neural network where word embedding will be formed after the training. Finally, with the use of PCA, a better representation will be made in terms of their reparability.

Before Training:

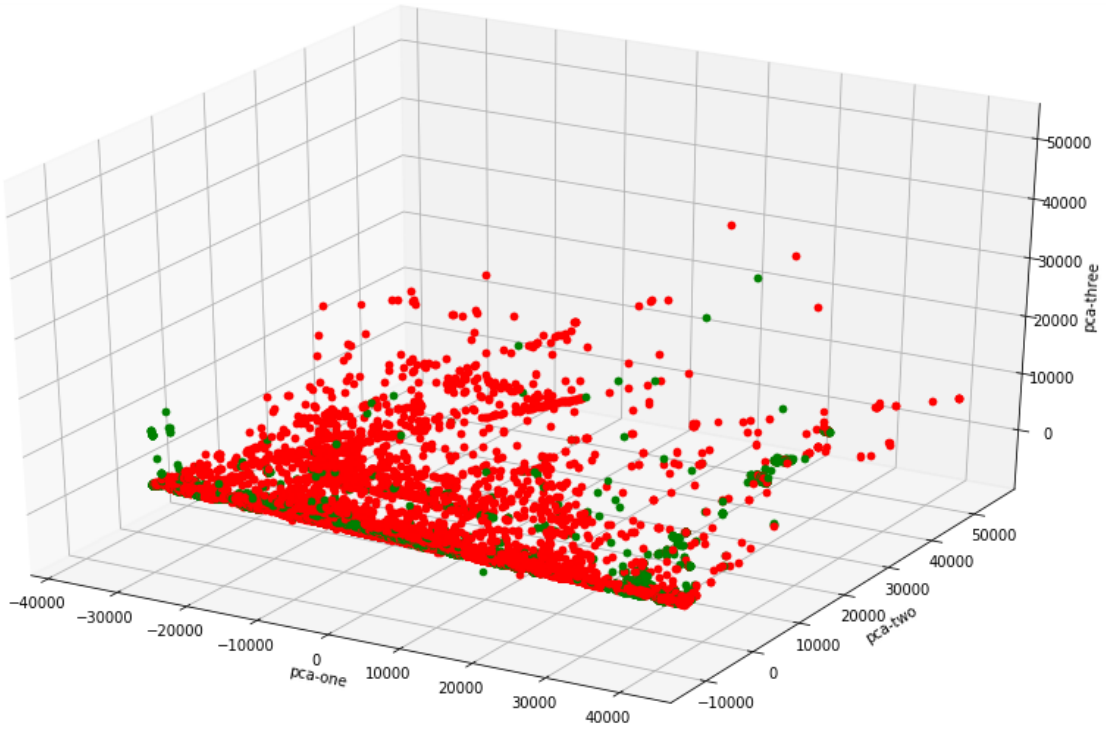


Figure 5.2: Cancer Variants in 3D before training in neural network (Red bullets means "Cancer Variants", Green bullets means "Likely Benign Cancer Variants").

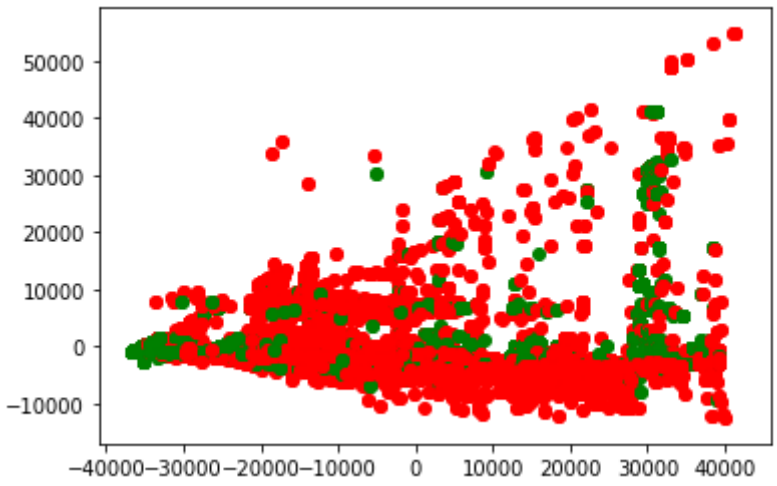
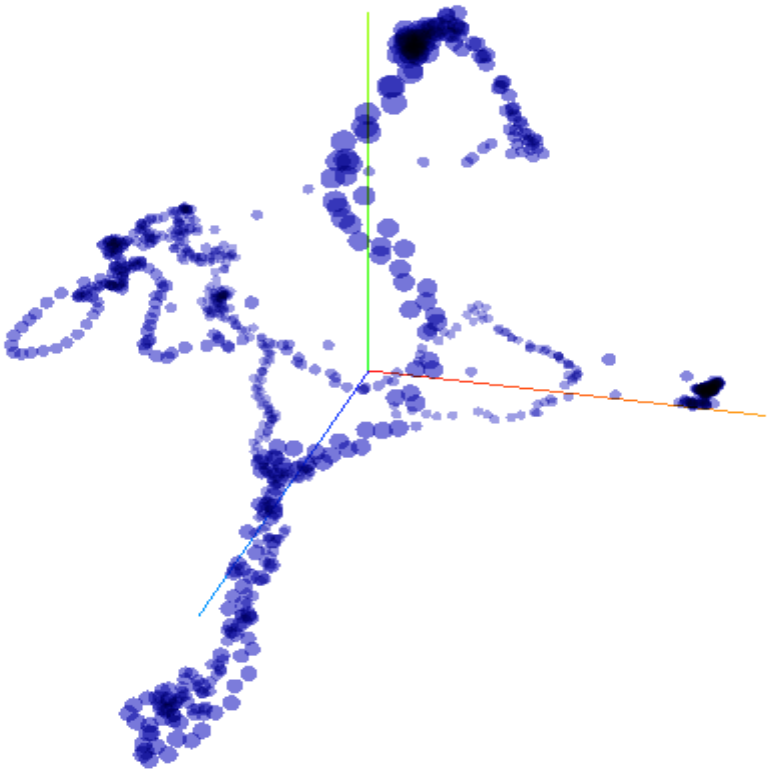


Figure 5.3: Cancer Variants in 2D PCA plot before training in neural network (Red bullets means "Cancer Variants", Green bullets means "Likely Benign Cancer Variants").

After Training:



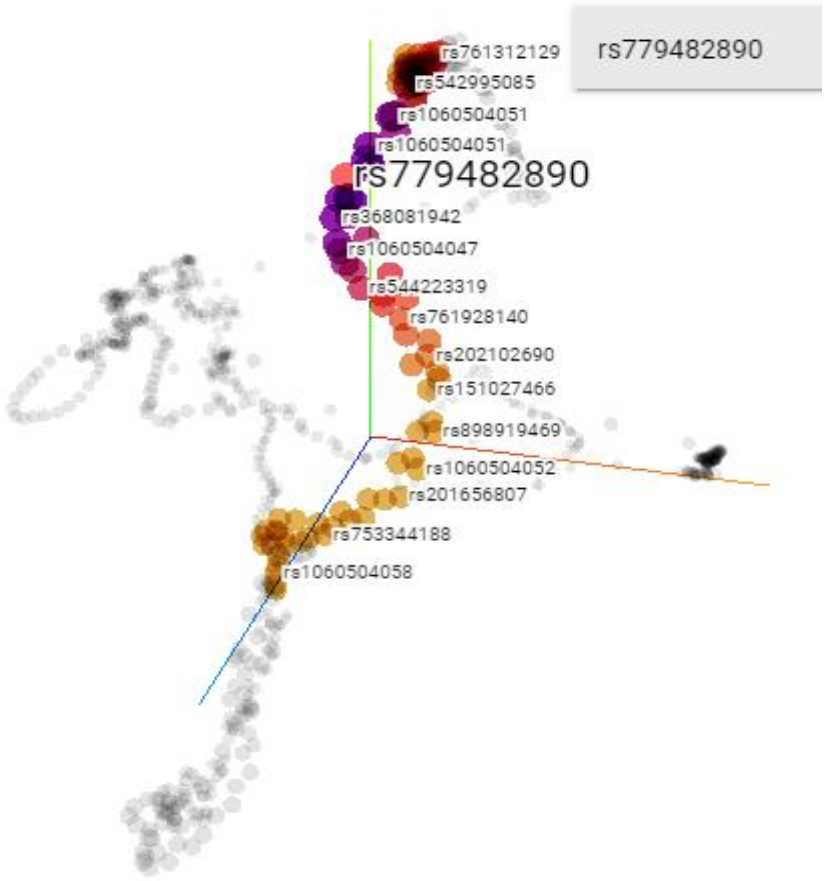


Figure 5.4: Embedding Projection of 900 Cancer SNPs and Likely Being Cancer SNPs.

In Figure 5.2 and 5.4 we noticed the embedding results after the training of our model (150 epochs ≈ 5 days) with 900 SNPs, 900 dimensions and variation data of two categories "Cancer" and "Likely being cancer". As an evaluation, we choose a random SNP and we search and watch the closest SNP's using Cosine Similarity. Then searching in the literature and genomics databases we try to find evidence for the similarity or common ground for these two SNPs in terms of disease. The result seems to be accurate because the data was fewer than Experiment 1 and have class. The total variance description rate is 61.2% (3rd Dimension) on figure 5.4 according to TensorFlow projector. Also, the colours represent how much close is a target SNP with others by a query using Cosine Similarity.

After Isolation:

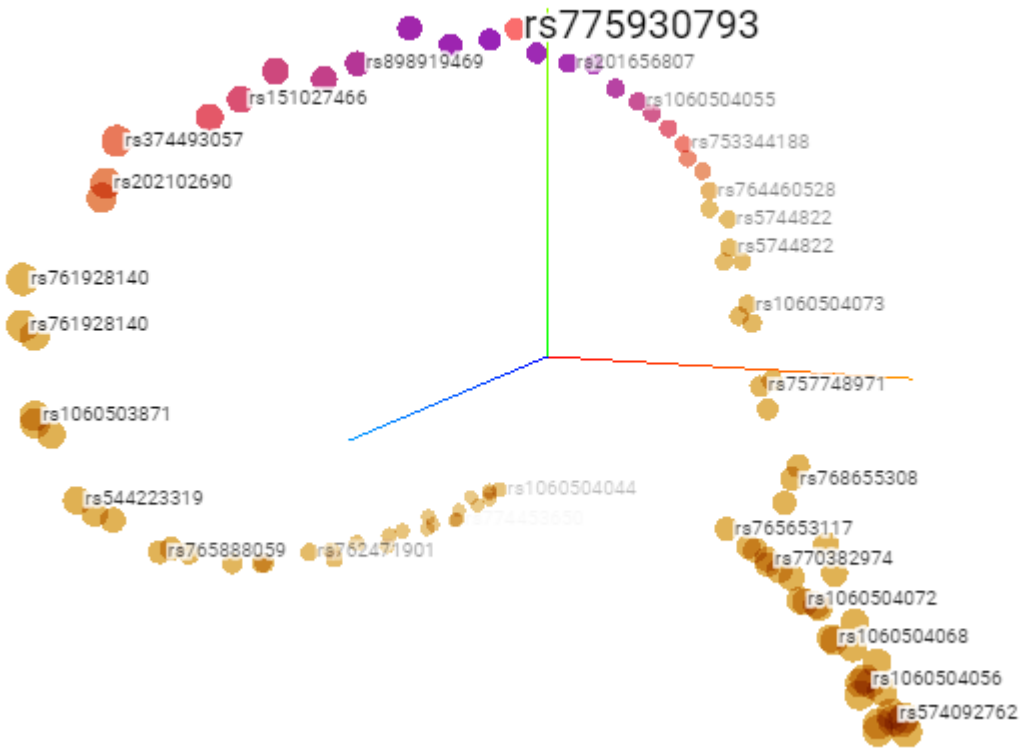


Figure 5.5: Embedding Projection of 100 Isolated Cancer SNPs and Likely Being Cancer SNPs (3D).

In Figure 5.5 we notice after using isolate in TensorFlow Projector that the total variance description was increased on 77.5% to a specific region based of the cosine similarity of one random chosen SNP. Also the dataset had some duplicate SNP's with some variations on START and END fields. In the figure above, it proves that the model works by putting them side by side the same contexts. And we guess that remains to find the SNP key that separates the "potentially cancerous" SNPs from the "cancerous". To do that we must check where are concentrated the most of cancerous variants and where "potentially cancerous variants" until guess semantically the category of each. The logic of some embedding it's like the roots of a tree. Based on some experiments such as the "Word Embedding Visual Inspector".

6 Third Experiment

6.1 Problem definition

On the third experiment, the objective is to categorize a SNP dataset to a specific disease. According to dataset where is selected by the criteria if exists a cancerous SNP. An expert in the field of bioinformatics provided to us a list of mutations that fall into one of the three following categories LIKELY_BENIGN: 11119, BENIGN: 3547 and PATHOGENIC: 14499 - TOTAL: 29165 SNP's. We discovered after using Principal Component Analysis (PCA) and we plot it, that it's hard to cluster only "cancer" and "possibly cancer" variants and preview all those datasets by their unique attributes.

Another challenge after plotting those cancerous SNPs is to verify the results and somehow be able to find if a mutation has some correlation (or the opposite) with another one and if they are similar so that we can study them and possibly find new conclusions about these (e.g. a patient with mutation rs1234 has a predisposition to cancer).

According to the architecture we introduced in the methodology chapter we believe a deep learning neural model and more specific a natural language processing deep neural model will be able to help us analyses our data while the PCA to plot this type of data. Such a model is designed to process documents and produce vectors of words to be able to plot them to find the similarities between them. Again in the third experiment, we will try to use variant data instead of words to be able to produce vectors of variants.

6.2 Data

Some of these fields in table below will be used in the neural network as input. Those fields are a basic description of a mutation.

VariantType
Gene
RS
Chr
Ref_Allele
Alt_Allele
Cancer_MedGen_Pheno
Classification

Table 6.1: Cancer Variants Attributes

6.3 Results

In the third experiment, we have a database of cancerous SNPs and possibly cancerous. This data has many similarities in their information and consist of a multidimensional structure. To begin with, it was analyzed by PCA, resulting in the observation that the sample is quite homogeneous in the first phase. Then, contexts and labels where created for these groups ("Cancer Variants" and "Likely Benign Cancer Variants" group). The need to separate our dataset in groups for a better experience in visualization. The context (positions of attributes) and labels (RS) where that are the input for the neural network where word embedding will be formed after the training. Finally, with the use of PCA, a better representation will be made in terms of their reparability.

After Training:

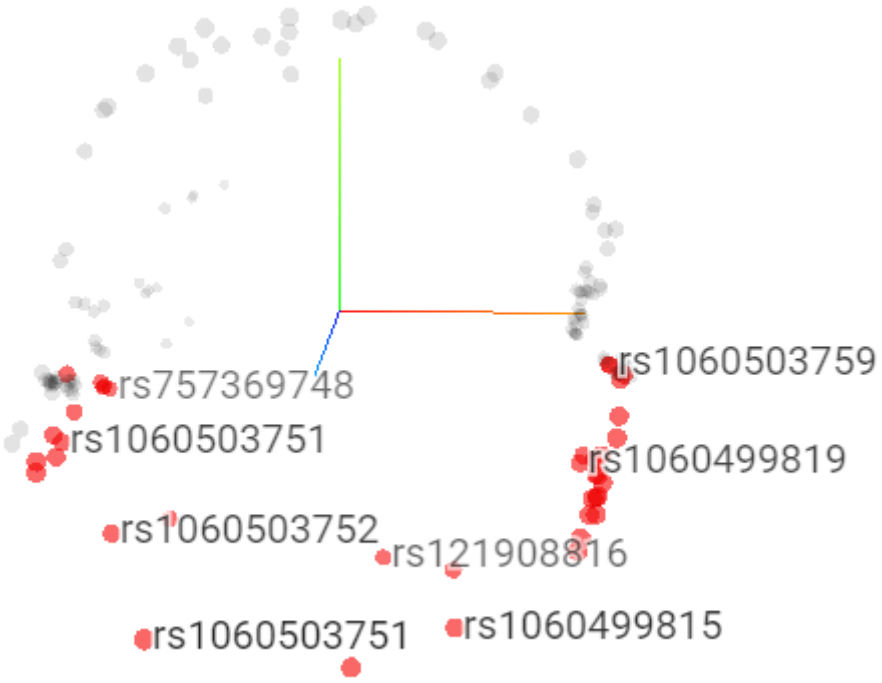


Figure 6.2: Embedding Projection of 143 Cancer SNPs and Likely Being Cancer SNPs (Red bullets are cancer SNPs).

In Figure 6.2 we noticed the embedding results after the training of our model (118 epochs ≈ 8 hours) with 143 SNPs, 143 dimensions and variation data of two categories "Cancer" and "Likely being cancer". The result seems to be accurate because the data was fewer than Experiment 1 and have class as attribute and visualization taxonomy. The total variance description rate is 95.5% (3rd Dimension).

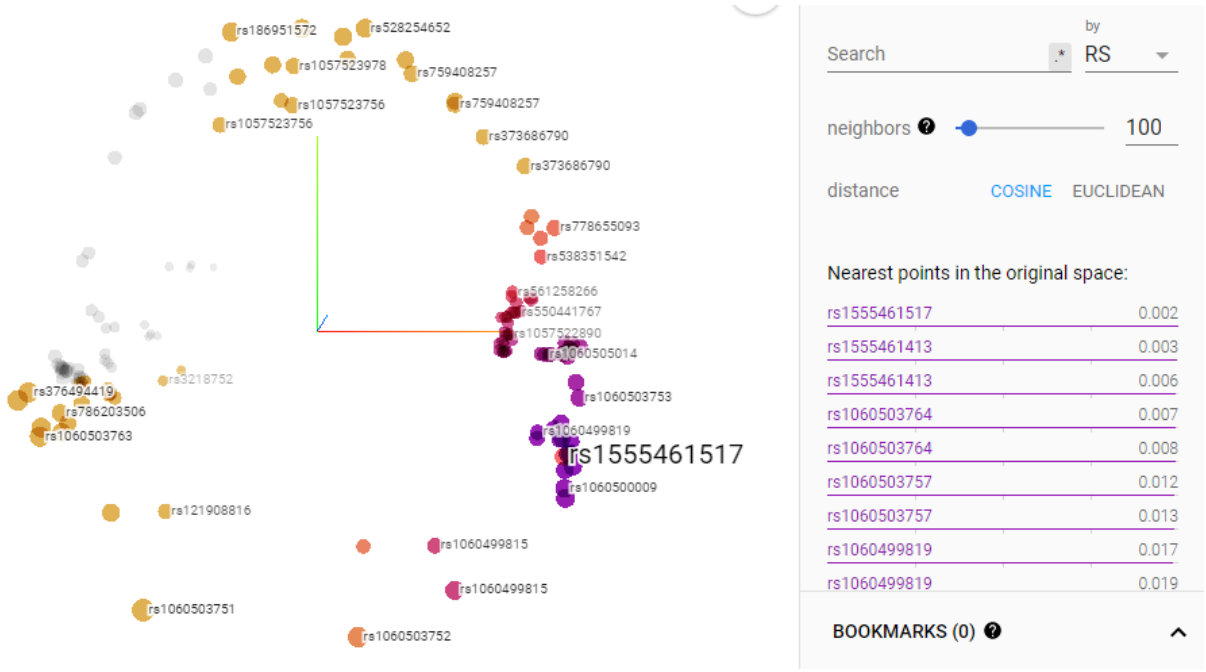


Figure 6.3: Embedding Projection of one random chosen SNP from 143 Cancer SNPs and Likely Being Cancer SNPs.

In Figure 6.3 as an evaluation, we choose a random SNP and we search and watch the closest SNP's using Cosine Similarity. The colours represent how much close is a target SNP with others by a query using Cosine Similarity. Then searching in the literature and genomics databases we try to find evidence for the similarity or common ground for these two SNPs in terms of disease. We notice in TensorFlow Projector that the dataset had some duplicate SNP's with some variations on START and END fields (OR data from another variants database). In the figure above, it proves that the model works by putting them side by side the same contexts. The next step is to verify some of the similarities of rs1555461517.

No	SNPs	ATTRIBUTES
1	rs1555461517	['duplication' 'PALB2' 'rs1555461517' '16' '23635570' '23635570' 'A' 'AA' 'criteria provided, multiple submitters, no conflicts' '2' 'C0006142' 'Pathogenic']
2	rs1555461413	['duplication' 'PALB2' 'rs1555461413' '16' '23635360' '23635360' 'A' 'AA' 'criteria provided, multiple submitters, no conflicts' '2' 'C0006142' 'Pathogenic']
3	rs1060503764	['duplication' 'SDHB' 'rs1060503764' '1' '17349151' '17349151' 'A' 'AA' 'criteria provided, multiple submitters, no conflicts' '2' 'C0031511' 'Pathogenic']

Table 6.4: Compare variants from our dataset.

According table 6.4 the result seems valid I choose the SNP *rs1555461517* as target and the results is like a chain.

The *rs1555461517* (no. 1) with *rs1555461413* (no. 2) they are almost the same. Specific the:

- VariantType = ‘duplication’
- Gene = ‘PALB2’
- Chromosome = 16
- Ref_Allele = ‘A’
- Alt_Allele = ‘AA’
- Cancer_MedGen_Pheno = ‘C0006142’
- Category = ‘Pathogenic’ (Cancer).

The no. 1,2 shares with no.3:

- VariantType = ‘duplication’
- Ref_Allele = ‘A’,
- Alt_Allele = ‘AA’
- Category = ‘Pathogenic’ (Cancer).

7 Discussion

The thesis aimed to represent and compare mutations in the human genome using deep learning methodologies. Deep learning is an AI discipline that the last years have gained great momentum in many research areas. One of the areas is the bioinformatics where we can find methodologies for gene expression inference [38] response to therapy [39], predicts missing methylation states [40] and many other in the fields of gene signatures, pathway analysis [41][42] and radiogenomics [43][44]. To our knowledge, this is the first study that tries to map the human mutations as a text mining problem where the DNA sequencing per sample/human can act as a document and the known mutations as words of this imaginary vocabulary. There is a lot of research in the field of natural language processing for word prediction and the last years' solutions that take advantage of deep learning methods have proven to provide much better results that traditional method [45][46].

NLP methods can assist us in predicting protein activity after found a valid SNP from experiments, wherein bioinformatics is one of the most basic research topics. It includes the localization of subcellular protein [47], prediction of protein to protein interactions and interaction sites [48], the detection of protein remote homology, the classification of protein functions, the prediction and classification of a transmembrane protein, the recognition of multifunctional enzymes and the identification of DNA binding protein. [37]

In this dissertation, we did three experiments to analyze human DNA and find mutations. Regarding the first experiment, the desired result was to produce the appropriate vectors to highlight the best representation of the mutations. We conclude that the volume of data greatly affects the processing time of the results in a long delay. This affects the validity of the results because it will take months for the validity of the results to be able to verify and compare the mutations according to the model used. Also, for this particular experiment, an analysis was made only for chromosome 22, which shows us that for a multidimensional mutation that will target more chromosomes, the diagnosis will be more complex and time-consuming.

For the realization of the second experiment we followed the almost the same procedure with the first experiment with the exception that the input is encoded in one-hot and not implemented in Tensorflow. We concluded that the results given could be said to be quite valid by classification criteria. The improved accuracy can be attributed to the reduced input used instead of the whole number of mutations to two specific categories of carcinogens and potential carcinogens. Using categorization, we concluded that the data in our experiment due to the training of the neural model (150 epochs) and the production of vectors can become more efficient operations and find inhomogeneity. An important part that facilitated this process is that the volume of data is now smaller and the training of the model used required a shorter period of time than the first experiment performed. This provided a solution to the above experiment for which its main problem was the long processing time due to the oversized volume of information.

Again, with the realization of the third experiment which followed the same procedure with the first experiment, we concluded that the results given could be said are valid. Using this technique, we concluded that the data in our experiment due to the training of the neural model (118 epochs) and the production of vectors can provide valuable results. An important part that facilitated this process is that the volume of data is now smaller and the training of the model used required a shorter period of time than the first experiment performed. This provided a solution to the first experiment for which its main problem was the long processing time due to the oversized volume of information. Thus, according to the first experiment, it is understood that the categorization and focus on specific groups of mutations result in the export of faster and more valid results so that verification can be performed in a shorter period of time.

8 Conclusions

This section presents a brief synopsis of the dissertation, assessing major points of the implemented model and experiments. After all, we conclude by mentioning some beneficial improvements and extensions that could be developed in the near future.

To begin with, the need for identification of differences between variants for a better understanding of the human mutations is the key-point of the specific dissertation. To deal with such a problem, specific data selected as input to our deep learning model. This data was generated through Ensembl Variant Effect Predictor (VEP) [1] giving us the mutations that are taking place. At the next step, we preprocess the mutation data and transform them into numbers and choose the most valuable information about the variants. The main criteria for this selection are the uniqueness, statistics, the correlation of attributes and of course the contribution of bioinformaticians and biologists. Those attributes were used as input to a Continues Bags of Words Model (CBOW) [4] [49], and NLP model that gives us the ability to use as input multi-words contexts. When computing the hidden layer output, the CBOW model takes the average of the vectors of the input context words and use the average vector as the output. That model allows us to cluster N number of SNPs by their correlation since it uses the contexts as input and as target all the SNP's. Those vectors, also called word embeddings, used as input to PCA [30] to calculate the principal components (2D or 3D), and visualize our results. After that analysis will see and understand the general picture of variants and the differences between them.

It could also be argued that with the implementation of the first experiment we concluded that the volume of data affects the data processing time resulting in a long delay in the creation of a representation. Also, for this particular experiment, an analysis was made only for chromosome 22 were described by 1.083.881 variants, which highlight that for an extended analysis with more chromosomes, the diagnosis will be more complex and time-consuming. Then with the completion of the second experiment which followed the same procedure as the first experiment, we concluded that the results given could be said to be quite valid. The reason is that instead of using the whole number of mutations we limited to two specific categories (carcinogens and potential carcinogens). Using categorization, we concluded that the model in our second and third experiment can produce more efficient operations and find inhomogeneity. Furthermore, an important part that facilitated this process is that the volume of data is now smaller and the training of the model used required less time than the first experiment.

Unfortunately, there can be no doubt that the problem of optimization in TensorFlow version 2.X remains. Taking into consideration this problem, future research about the optimization and the training time. Moreover, another research in this area could be done to optimize CBOW and prevent the use of target contexts. Last but not least, results presentation improvements could be made, where a program could be created using an API in SNP databases (e.g., MCBI) where the software would compare an SNP with the nearest SNPs and some from the fields described in the database. The user will increase the success rate of our model so that we can make a second evaluation of it beyond the training.

Upon completion of the research, the essential conclusions are that the human genome consists of a chaotic set of information in which the method we followed confirmed both the difficulty and the complexity of analyzing this information. For this reason, categorizing and

focusing on a smaller amount of information can give us specific information relatively more effectively and clarifying, giving us a more meaningful picture.

9 Appendix

9.1 Formula of PCA

9.1.1 Example: From 3-Dim to 2-Dim with Principal Component Analysis

Imagine that you have a 3-dimensional dataset A with 5 row and 3 columns. Where every dimension named x, y and z.

$$A = \begin{bmatrix} 1.5 & 2.4 & 1.9 \\ 2.0 & 0.7 & 1.2 \\ 1.2 & 2.9 & 4.9 \\ 1.4 & 2.2 & 4.2 \\ 5.1 & 3.0 & 5.0 \end{bmatrix}$$

The mean for matrix A is:

$$Mean_A = \begin{bmatrix} (1.5 - \bar{x}) = -0.94 & (2.4 - \bar{y}) = 0.16 & (1.9 - \bar{z}) = -1.54 \\ (2.0 - \bar{x}) = -0.44 & (0.7 - \bar{y}) = -1.54 & (1.2 - \bar{z}) = -2.24 \\ (1.2 - \bar{x}) = -0.24 & (2.9 - \bar{y}) = 0.66 & (4.9 - \bar{z}) = 1.46 \\ (1.4 - \bar{x}) = -1.04 & (2.2 - \bar{y}) = -0.04 & (4.2 - \bar{z}) = 0.76 \\ (5.1 - \bar{x}) = 2.66 & (3.0 - \bar{y}) = 0.76 & (5.0 - \bar{z}) = 1.56 \end{bmatrix}$$

The covariance for $Mean_A$:

$$cov(Mean_A) = \begin{bmatrix} conv(x,x) & conv(x,y) & conv(x,z) \\ conv(y,x) & conv(y,y) & conv(y,z) \\ conv(z,x) & conv(z,y) & conv(z,z) \end{bmatrix}$$

$$cov(x,x) = \ var(x) = \frac{(-0.94)^2 + (-0{,}44)^2 + (-0.24)^2 + (-1.04)^2 + (2.66)^2}{(5-1)} = 2.3230$$

$$cov(y,y) = var(y) = \frac{(0.16)^2 + (-1.54)^2 + (0.66)^2 + (-0.04)^2 + (0.76)^2}{(5-1)} = 0.8530$$

$$cov(z,z) = var(z) = \frac{(-1.54)^2 + (-2.24)^2 + (1.46)^2 + (0.76)^2 + (1.56)^2}{(5-1)} = 3.1330$$

$$\begin{aligned} cov(x,y) &= cov(y,x) \\ &= \frac{1}{(5-1)}[(-0.94 - \bar{x})(0.16 - \bar{y}) + (-0.44 - \bar{x})(1.54 - \bar{y}) \\ &+ (-0.24 - \bar{x})(0.66 - \bar{y}) + (-1.04 - \bar{x})(-0.04 - \bar{y}) \\ &+ (2.66 - \bar{x})(0.76 - \bar{y})] = 0.6080 \end{aligned}$$

$$\begin{aligned} cov(x,z) &= cov(z,x) \\ &= \frac{1}{(n-1)}[(-0.94 - \bar{x})(-1.54 - \bar{z}) + (-0.44 - \bar{x})(-2.24 - \bar{z}) \\ &+ (-0.24 - \bar{x})(1.46 - \bar{z}) + (-1.04 - \bar{x})(0.76 - \bar{z}) + (2.66 - \bar{x})(1.56 - \bar{z})] \\ &= 1.3605 \end{aligned}$$

$$cov(y,z) = cov(z,y)$$
$$= \frac{1}{(n-1)}[(0.16-\bar{y})(-1.54-\bar{z}) + (-1.54-\bar{y})(-2.24-\bar{z})$$
$$+ (0.66-\bar{y})(1.46-\bar{z}) + (-0.04-\bar{y})(0.76-\bar{z}) + (0.76-\bar{y})(1.56-\bar{z})$$
$$= 1.3305$$

$$cov_A = \begin{bmatrix} 2.3230 & 0.6080 & 1.3605 \\ 0.6080 & 0.8530 & 1.3305 \\ 1.3605 & 1.3305 & 3.1330 \end{bmatrix}$$

$$det\left(\begin{bmatrix} 2.3230 & 0.6080 & 1.3605 \\ 0.6080 & 0.8530 & 1.3305 \\ 1.3605 & 1.3305 & 3.1330 \end{bmatrix} - \lambda \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}\right)$$
$$= -\lambda^3 + 6{,}309\lambda^2 - 7{,}9410725\lambda + 1{,}559964375$$

Now we must solve: $-\lambda^3 + 6{,}309\lambda^2 - 7{,}9410725\lambda + 1{,}559964375 = 0$

You can use Newton-Raphson method to find roots with formula:

$$X_n + 1 = X_n - \frac{f(X_n)}{f'(X_n)}$$

So we found 3 Eigen Values:

$$\lambda \approx 0.24072$$

$$\lambda \approx 1.38316$$

$$\lambda \approx 4.68510$$

And Eigen Vectors are:

$$eig1 = (cov\_A - \lambda I) = \begin{pmatrix} 2.0827 & 0.6080 & 1.3605 \\ 0.6080 & 0.61227 & 1.3305 \\ 1.3605 & 1.3305 & 2.89227 \end{pmatrix}$$

Now let's reduce the matrix to this form: $\begin{pmatrix} a & \cdots & b \\ 0 & \ddots & \vdots \\ 0 & 0 & c \end{pmatrix}$

$$= \begin{pmatrix} 2.0827 & 0.6080 & 1.3605 \\ 0.6080 & 0.61227 & 1.3305 \\ 1.3605 & 1.3305 & 2.89227 \end{pmatrix} \xrightarrow{R2 \leftarrow R2 - 0.29198 \cdot R1} \begin{pmatrix} 2.0827 & 0.6080 & 1.3605 \\ 0 & 0.43474 & 0.93324 \\ 1.3605 & 1.3305 & 2.89227 \end{pmatrix}$$

$$\xrightarrow{R3 \leftarrow R3 - 0{,}65337 \cdot R1} \begin{pmatrix} 2.0827 & 0.6080 & 1.3605 \\ 0 & 0.43474 & 0.93324 \\ 0 & 0.93324 & 2.00336 \end{pmatrix} \xrightarrow{R2 \leftrightarrow R3} \begin{pmatrix} 2.0827 & 0.6080 & 1.3605 \\ 0 & 0.93324 & 2.00336 \\ 0 & 0.43474 & 0.93324 \end{pmatrix}$$

$$\xrightarrow{R3 \leftarrow R3 - 0{,}46584 \cdot R2} \begin{pmatrix} 2.0827 & 0.6080 & 1.3605 \\ 0 & 0.93324 & 2.00336 \\ 0 & 0 & 0 \end{pmatrix}$$

Reduce matrix to this form $\begin{pmatrix} 1 & \cdots & b \\ 0 & \ddots & \vdots \\ 0 & 0 & 1 \end{pmatrix}$

$$\begin{pmatrix} 2.0827 & 0.6080 & 1.3605 \\ 0 & 0.93324 & 2.00336 \\ 0 & 0 & 0 \end{pmatrix} \xrightarrow{R2 \leftarrow 1.07152 \cdot R2} \begin{pmatrix} 2.0827 & 0.6080 & 1.3605 \\ 0 & 1 & 2.14665 \\ 0 & 0 & 0 \end{pmatrix}$$

$$\xrightarrow{R1 \leftarrow R1 - 0.608 \cdot R2} \begin{pmatrix} 2.0827 & 0 & 0.05533 \\ 0 & 1 & 2.14665 \\ 0 & 0 & 0 \end{pmatrix} \xrightarrow{R1 \leftarrow 0.48024 \cdot R1} \begin{pmatrix} 1 & 0 & 0.02657 \\ 0 & 1 & 2.14665 \\ 0 & 0 & 0 \end{pmatrix}$$

The system with Eigen value λ≈0.24072:

$$(A - 0.24072 \cdot I)\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0.02657 \\ 0 & 1 & 2.14665 \\ 0 & 0 & 0 \end{pmatrix} \cdot \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$$

$$\begin{matrix} x + 0.02657z = 0 \\ y + 2.14665 + z = 0 \end{matrix} \} = \begin{matrix} x - 0.02657z = 0 \\ y - 2.14665 + z = 0 \end{matrix} \} = \begin{pmatrix} -0.0265 \\ -2.14665 \\ 1 \end{pmatrix}$$

We do the same to find the Eigen vector 1 and Eigen vector 2. So:

$$\text{eig2} = \begin{pmatrix} -1.76308 \\ 0.48766 \\ 1 \end{pmatrix} \text{eig3} = \begin{pmatrix} 0.69366 \\ 0.45725 \\ 1 \end{pmatrix}$$

Next we must sort all Eigen vectors, as Inversed Feature Vector (Descending Sort):

$$Feature\ Vector = \begin{bmatrix} 0.53354838 & 0.3517079 & 0.76917337 \\ 0.84569511 & -0.23391879 & -0.47966842 \\ -0.01122093 & -0.90641246 & 0.42224464 \end{bmatrix}$$

$$PCA = ORIGINAL\_DATA^T\ x\ Feature\_Vector$$

And the dot product is:

PC1	**PC2**
-1.629789	-0.093691
-2.499340	1.062586
1.227069	-1.057669
0.015613	-1.234714
2.886447	1.323488

[30]

10 Bibliography

[1] W. Mclaren *et al.*, “The Ensembl Variant Effect Predictor,” 2016.

[2] L. Koumakis, “Deep learning models in genomics; are we there yet?,” *Computational and Structural Biotechnology Journal*, vol. 18. Elsevier B.V., pp. 1466–1473, 01-Jan-2020.

[3] S. Min, B. Lee, and S. Yoon, “Deep Learning in Bioinformatics.”

[4] X. Rong, “word2vec Parameter Learning Explained,” 2016.

[5] N. Anas *et al.*, “A review of human genome project (HGP) from ethical perspectives,” *Artic. Int. J. Adv. Appl. Sci.*, vol. 4, no. 12, pp. 125–132, 2017.

[6] L. Luzzatto, “Sickle Cell Anaemia and Malaria,” *Open J. Syst. Mediterr. J. Hematol. Infect. Dis. Cit. Mediterr J Hematol Infect Dis*, vol. 4, no. 1, p. 2012065, 2012.

[7] G. E. Hoffman, J. Bendl, K. Girdhar, E. E. Schadt, and P. Roussos, “Functional interpretation of genetic variants using deep learning predicts impact on chromatin accessibility and histone modification,” *Nucleic Acids Res.*, vol. 47, pp. 10597–10611, 2019.

[8] R. Poplin *et al.*, “A universal snp and small-indel variant caller using deep neural networks,” *Nat. Biotechnol.*, vol. 36, no. 10, p. 983, 2018.

[9] J. Zhou and O. G. Troyanskaya, “Predicting effects of noncoding variants with deep learning-based sequence model,” vol. 12, no. 10, 2015.

[10] D. S. Latchman, “Transcription factors: an overview Function of transcription factors,” 1993.

[11] F. Molnár, “Histone Modifications Human Epigenomics View project.”

[12] A. P. Boyle *et al.*, “High-Resolution Mapping and Characterization of Open Chromatin across the Genome.”

[13] T. Mikolov, K. Chen, G. Corrado, and J. Dean, “Efficient Estimation of Word Representations in Vector Space.”

[14] J. Zhang, “Basic Neural Units of the Brain: Neurons, Synapses and Action Potential.”

[15] IBM, “The Neural Networks Model,” 2012. [Online]. Available: https://www.ibm.com/support/knowledgecenter/en/SS3RA7_15.0.0/com.ibm.spss.modeler.help/neuralnet_model.htm. [Accessed: 07-Feb-2020].

[16] I. H. Witten, E. Frank, and M. A. Hall, *Data Mining*. 2011.

[17] B. Kim, H. Kim, K. Kim, S. Kim, and J. Kim, “Learning Not to Learn: Training Deep Neural Networks with Biased Data.”

[18] V. Zhou, “Machine Learning for Beginners: An Introduction to Neural Networks,” 2019. [Online]. Available: https://towardsdatascience.com/machine-learning-for-beginners-an-introduction-to-neural-networks-d49f22d238f9. [Accessed: 07-Feb-2020].

[19] W. Fan, L. Wallace, and Z. Zhang, “Tapping the Power of Text Mining Text Analytics

View project Customer Agility View project," *Artic. Commun. ACM*, 2006.

[20] S. H. Liao, P. H. Chu, and P. Y. Hsiao, "Data mining techniques and applications - A decade review from 2000 to 2011," *Expert Syst. Appl.*, vol. 39, no. 12, pp. 11303–11311, 2012.

[21] R. Talib, M. K. Hanif, S. Ayesha, and F. Fatima, "Text Mining: Techniques, Applications and Issues," 2016.

[22] Y. Wilks, "Natural Language Processing," *Commun. ACM*, vol. 39, no. 1, pp. 60–62, 1996.

[23] G. G. Chowdhury, "Natural language processing," 2003.

[24] Y. Goldberg *et al.*, "Word2Vec Explained: Deriving Mikolov et al.'s Negative-Sampling Word-Embedding Method," 2014.

[25] B. Jangid, I. Kim, and J. W. Kim, "Word2vec convolutional neural networks for classification of news articles and tweets," 2019.

[26] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Distributed Representations of Words and Phrases and their Compositionality."

[27] R. Collobert, J. Weston, J. Com, M. Karlen, K. Kavukcuoglu, and P. Kuksa, "Natural Language Processing (Almost) from Scratch," 2011.

[28] W. Ling, I. Trancoso, C. Dyer, and A. Black, "Two/Too Simple Adaptations of Word2Vec for Syntax Problems," 2015.

[29] "CS 224D: Deep Learning for NLP 1."

[30] U. Sarkar, S. Taraphder, and S. Datta, "Principal Component Analysis," 2017.

[31] T. Kitasuka, M. Aritsugi, and F. Rahutomo, "Semantic Cosine Similarity," 2012.

[32] H. Zwart, "Human Genome Project: History and Assessment," 2015.

[33] "Human Variation Sets in VCF Format." [Online]. Available: https://www.ncbi.nlm.nih.gov/variation/docs/human_variation_vcf/. [Accessed: 18-Dec-2020].

[34] "VEP Data formats." [Online]. Available: https://www.ensembl.org/info/docs/tools/vep/vep_formats.html. [Accessed: 20-Aug-2020].

[35] B. Wang, A. Wang, C. Fenxiao, Y. Wang, and C.-C. J. Kuo, "Evaluating Word Embedding Models: Methods and Experimental Results."

[36] L. Lovmar, A. Ahlford, and A.-C. Syvänen, "Silhouette scores for assessment of SNP genotype clusters," 2005.

[37] Z. Zeng, H. Shi, Y. Wu, and Z. Hong, "Survey of Natural Language Processing Techniques in Bioinformatics," 2015.

[38] Y. Chen, Y. Li, R. Narayan, A. Subramanian, and X. Xie, "Gene expression inference with deep learning," *Bioinformatics*, vol. 32, no. 12, pp. 1832–1839, Jun. 2016.

[39] T. Sakellaropoulos *et al.*, "A Deep Learning Framework for Predicting Response to Therapy in Cancer," *Cell Rep.*, vol. 29, no. 11, pp. 3367-3373.e4, Dec. 2019.

[40] C. Angermueller, H. J. Lee, W. Reik, and O. Stegle, "DeepCpG: Accurate prediction of single-cell DNA methylation states using deep learning," *Genome Biol.*, vol. 18, no. 1, p. 67, Apr. 2017.

[41] L. Koumakis, V. Moustakis, M. Zervakis, D. Kafetzopoulos, and G. Potamias, "Coupling regulatory networks and microarays: revealing molecular regulations of breast cancer treatment responses." Springer, Berlin, Heidelberg, pp. 239–246, 2012.

[42] L. Koumakis *et al.*, "MinePath: mining for phenotype differential sub-paths in molecular pathways," *PLoS computational biology 12*, vol. 11, PLoS, 2016.

[43] Mehta, Shaveta *et al.*, "Radiogenomics monitoring in breast cancer identifies metabolism and immune checkpoints as early actionable mechanisms of resistance to anti-angiogenic treatment." EBioMedicine 10, pp. 109–116, 2016.

[44] E. Trivizakis *et al.*, "Artificial intelligence radiogenomics for advancing precision and effectiveness in oncologic care.," *Int. J. Oncol. 57*, vol. 1, pp. 43–53, 2020.

[45] L. Zhang, S. Wang, and B. Liu, "Deep learning for sentiment analysis: A survey.," *Wiley Interdiscip. Rev. Data Min. Knowl. Discov. 8*, vol. 4, no. e1253, 2018.

[46] V. Menger, F. Scheepers, and M. Spruit, "Comparing deep learning and classical machine learning approaches for predicting inpatient violence incidents from clinical text.," *Appl. Sci. 8*, vol. 6, no. 981, 2018.

[47] Z. Wang, Q. Zou, Y. Jiang, Y. Ju, and X. Zeng, "Review of Protein Subcellular Localization Prediction," *Curr. Bioinform.*, vol. 9, no. 3, pp. 331–342, Sep. 2014.

[48] B. Liu, X. Wang, L. Lin, and Q. Dong, "Exploiting three kinds of interface propensities to identify protein binding sites," *Comput. Biol. Chem. no.4*, vol. 33, pp. 303–311, 2009.

[49] S. Kuang, B. D. Davison, S. Kuang, and B. D. Davison, "Learning class-specific word embeddings."