UNIVERSITY OF CALIFORNIA,
IRVINE

Colorectal cancer predictive test using germ-line DNA data and multiple machine learning methods

DISSERTATION

submitted in partial satisfaction of the requirements
for the degree of

DOCTOR OF PHILOSOPHY

in Biomedical Engineering

by

Buyun Zhang

Dissertation Committee:
Associate Professor James P. Brody, Chair
Assistant Professor Michelle Digman
Assistant Professor Jered Haun

2019

DEDICATION

To

my parents and friends

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

I would like to express the deepest appreciation to my committee chair, Professor James P. Brody, for his endless support and nurture throughout my graduate career. His supervising was creative, inspiring, and productive. Not only was he a great mentor in my studies, but he was also a good teacher in my life. His kindness guidance and patience often inspire me in every aspect of life. Without his support and persistent help, this dissertation would not have been possible.

I would like to thank my committee members, Professor Michelle Digman, and Professor Jered Haun, for continued support and devotion to this dissertation. Also, a thank you to my internship mentor at Facebook, Xianming Liu. His extensive knowledge broadened my horizon in machine learning studies.

Finally, I would like to thank my parents and friends, who have always been strong support in my life. Especially, my girlfriend Linting Sun, who has always been a constant source of inspiration.

CURRICULUM VITAE

Buyun Zhang

2015 B.S. in Information Engineering, Zhejiang University

2018 M.S. in Biomedical Engineering, University of California, Irvine

2018 Software Engineer Intern, Machine Learning, Facebook Inc

2019 Ph.D. in Biomedical Engineering, University of California, Irvine

FIELD OF STUDY

Bioinformatics and machine learning

PUBLICATIONS

"Artificial Intelligence" Buyun Zhang, and James P. Brody. Chapter14, Engineering-Medicine: Principles and Applications of Engineering in Medicine. CRC Press, 2019.

ABSTRACT OF THE DISSERTATION

Colorectal cancer predictive test using germ-line DNA data and multiple machine learning methods

By

Buyun Zhang

Doctor of Philosophy in Biomedical Engineering

University of California, Irvine, 2019

Associate Professor James P. Brody, Chair

Colorectal cancer is the third most common type of cancer in the world. Colorectal cancer begins with small, noncancerous clumps of cells (polyps). Early screening, with colonoscopies, and surgical removal of polyps can prevent the cancer from ever developing. However, colonoscopies are expensive and painful, and the lifetime risk of colon cancer is only 5%. A test to identify people who are likely to develop colon cancer could eliminate needless colonoscopies.

We obtained germline genetic data for 1309 patients diagnosed with colon cancer and compared this to 7517 others who have never been diagnosed with colon cancer. This dataset was collected as part of the Cancer Genome Atlas Program. We used supervised machine learning on this dataset to answer two questions. First, what fraction of these colon cancer patients should be predictable from germline data? Second, how well could such a test perform. We evaluated the performance of five different machine-

learning algorithms (gradient boost machine, wide neuron networks, deep neuron networks, dense-sparse neuron networks, and pairwise neuron networks) to answer these questions.

We found that about 78% of colon cancer cases in the dataset should be predictable from germline genetic data. We measured the receiver operating characteristic curve, which quantifies the tradeoff between sensitivity and specificity, for a germline genetic test that could predict a future diagnosis of colon cancer. We measured the area under the receiver operating characteristic curve to be about 0.80. We found that the gradient boost machine and pairwise neuron network algorithms perform equally well, and these two models were significantly better than the others. We conclude that a germline genetic test to predict a future diagnosis of colon cancer could be useful to focus screening on appropriate populations.

Chapter1 Introduction

Colorectal cancer is the third most common type of cancer. Around 1.4 million new cases are diagnosed annually, consisting of 10% of all cancer types(1). In the United States, the lifetime risk of colorectal cancer is estimated to be 4.49% for men and 4.15% for women, and it is expected to cause about 50,630 deaths during 2018.(2), (3)

Colorectal cancer originates with polyps in the colon or rectum. According to where these cancers start, they can also be named colon cancer or rectal cancer. Since colon cancer and rectal cancer have many features in common(4), they are grouped in many research reports.

The inner lining of the colon or rectum starts to grow when most colorectal cancers begin. These growths are called polyps. Most types of polyps are often non-cancerous growth, but some of them can develop into cancer(5). Two common types of polyps are found in the colon and rectum. The first type is hyperplastic and inflammatory polyps. They are the most common type of polyps and are not pre-cancerous in general. The second type is adenomas or adenomatous polyps. These polyps sometimes change into cancer. Therefore, adenomas are often called a pre-cancerous condition.

Around 96% of colorectal cancers are developed from adenocarcinomas(6). Some other sub-types of adenocarcinoma may have a worse prognosis such as signet ring and mucinous. Also, there are other less common types of tumors can start in the colon and

rectum. They are carcinoid tumors, gastrointestinal stromal tumors, lymphomas, and sarcomas(7). Carcinoid tumors start from special hormone-making cells in the intestine. Gastrointestinal stromal tumors start from special cells in the wall of the colon called the interstitial cells of Cajal. These tumors can be found anywhere in the digestive tract, but not common in the colon. Lymphomas mostly start in lymph nodes, but they can also begin in the colon. Sarcomas often start in connective tissues, and sarcomas in the colon or rectum are rare. Now, in the United States, colorectal cancer is the third most commonly diagnosed cancer among both men and women.

Colorectal cancer has many risk factors. They include environment factor and lifestyle factors such as smoking, heavy alcohol use, and type 2 diabetes. However, the most critical risk factors are the inherited factors according to previous studies(8). A family history of colorectal cancer or adenomatous polyps can increase the probability of getting colorectal cancer significantly. Nearly one in three people who develop colorectal cancer have other family members who have had it. The risk for people with a history of colorectal cancer in a first-degree relative (parent, sibling, or child) is also increased. The risk is even higher if that relative was diagnosed with colorectal cancer when they were younger than 45, or if more than one first-degree relative is affected.

Early diagnosis is one of the most effective ways to avoid or minimize psychological, physical, and financial suffering from colorectal cancer. However, early colonoscopy could be expensive and physical suffering. Environment (such as radiation and sunlight), lifestyle (such as diet and smoking), and inherited genetics are three primary carcinogenic

factors(9). One method of early diagnosis is a prediction from germline DNA. Genome-wide association studies (GWAS) are one of the most important studies which focus on the association between genetic variants and diseases. However, GWAS have not shown strong success in predicting the probabilities to common cancers (10,11). Among the previous studies, Single-nucleotide polymorphisms (SNPs) are one of the commonly used inherited factors(12,13,22–24,14–21).

However, germline SNPs analysis is unable to explain observed heritability thoroughly. All the common SNPs only explain 7.52% of the heritability(25). On the other hand, the largest twin study estimates the overall contribution of inherited genes to the development of colorectal cancer to be 35%, but it had a wide confidence interval (10%-48%)(26).

Copy number variation is also a critical inherited factor. It affects more base pairs per mutational events than SNPs. Copy number variation is a phenomenon in which the number of copies of a region of the genome varies between individuals in a human population. According to the HapMap Project, discrete copy number variations cover 12% of the human genome. The HapMap Project also indicate that CNVs encompass more polymorphic base pairs than SNPs(27). In this study, CNVs are used to be the major genetic factor in predicting colorectal cancer risk.

On the other hand, the complexity of the genomic factors also increased the difficulty of estimating the probability of getting colorectal cancer. Hence, developing new methods

of predicting the probability and estimating the predictability are necessary. Machine learning has the abilities to handle complex data, and it has wide application in bioinformatics. So, we choose machine learning as a tool for our study. To estimate the probability of getting colorectal cancer, the copy number variation data were obtained from patient blood samples, which were processed to extract germline DNA sequence. We built machine learning models to predict whether a person will develop colorectal cancer using germline DNA Copy Number Variation data. Also, a new method to estimate the predictability of colorectal cancer is reported.

Machine learning is a technique of computer science that uses statistical methods to give computer systems the ability to progressively improve performance with the existing data without being explicitly programmed(28). Computational statistics are closely related to machine learning, and it also focuses on prediction making using computational algorithms. The core idea of building a machine learning model is mathematical optimization. The efficiency of the optimization significantly affects the model quality.

Classification is one of the most important applications of machine learning. The prediction of getting colorectal cancer is a typical classification problem. In classification, the feature vectors (inputs) are the variables that are used to assign each case to certain classes. After training, the model could assign an unseen input to one of the classes. In our study, the machine learning model could estimate the probability of getting colorectal cancer and classify whether a person is likely to develop colorectal cancer or not.

In this study, multiple machine learning algorithms are used to predict colorectal cancer risk:

- Gradient Boost machine (GBM), a method of ensembling weak learners (simple machine learning algorithms, such as decision trees) into efficient learners. It has an excellent performance in many ranking and classification problems, such as the commercial web search engines in Yahoo(29).

- Neuron network is a complex computing system used to solve many practical problems such as classification. It has strong success in image classification, natural language processing, and recommendation systems recently. The architecture of neuron networks affects its performance significantly.

    - Wide networks are often used to simple and large scale problems.

    - Deep networks are widely used in many circumstances but sometimes suffer from the gradient exploding or gradient vanishing because of the large depth in the networks.

    - A dense-sparse network is developed to address the data sparsity problem and feature interactions in this study.

    - Also, a pairwise network is developed to push the predicted probabilities of colorectal patients and non-colorectal cancer patients away.

Chapter2 The principle of prediction and artificial intelligence in medicine

2.1 Introduction of machine learning and artificial intelligence

Artificial intelligence is a broad term that includes many different technologies and has a long history in medicine(30). In its broadest sense, artificial intelligence is defined as a machine that has thinking, reasoning, problem-solving, and learning capabilities similar to humans. Examples of artificial intelligence include knowledge bases and expert systems. Table 1 gives an example of an expert system for diagnosing congenital heart disease from 1968(31).

Table 1 demonstrates the early importance placed on natural language processing, the ability to extract information from written language. Natural language processing is still an area of active research with some success. However, today's most successful artificial intelligence applications in medicine use machine learning.

Table 1 An artificial intelligence package for diagnosis, 1968.

PLEASE TELL ME ABOUT THE PATIENT

The patient here with me is 23 years old

PLEASE CONTINUE

He shows no evidence of cyanosis

OK PLEASE GO ON

There is no indication of the squatting symptom

ALL RIGHT. GO ON PLEASE.

I'm afraid that's all I know about him at present.

CHECK THE HILAR VESSELS. ARE THEY INCREASED OR DECREASED

They are increased

I SEE

LOOK FOR A LEFT SECOND SYSTOLIC MURMUR

First tell me what the three most probable diseases are at this time

2 WITH PROBABILITY 74

18 WITH PROBABILITY 7

30 WITH PROBABILITY 3

PLEASE CONTINUE

LOOK FOR A LEFT SECOND SYSTOLIC MURMUR

And the name of disease 18 is

PATENT DUCTUS ARTERIOSUS

ALL RIGHT. GO ON PLEASE

LOOK FOR A LEFT SECOND SYSTOLIC MURMUR

He has a left second midsystolic murmur

ALL RIGHT

IS THE AXIS IN THE ECG RIGHT, LEFT, OR NORMAL

It is the right axis

OK

I RECOMMEND A DIAGNOSIS OF DISEASE 2

What is your current view of the problem

MY CURRENT VIEW IS

DISEASE	PROBABILITY
2	93
4	2

Name disease 4 for me

ATRIAL SEPTIC DEFECT WITH PULMONARY HYPERTENSION

I'm finished with this case. Thank you for your help

I HOPE THAT I HAVE BEEN OF SERVICE. GOODBYE

The recent popularity of machine learning can be attributed to two major innovations, first the availability of extremely large digitized datasets and second, the development of

computational techniques to control and reduce overfitting known as regularization. These large digital datasets are often called Big Data. Big Data has led to several innovations. Large collections of text that have been translated into two or more languages led to machine translations that are almost as good as humans(32). Thousands of hours of television programs that include closed captions for the hearing impaired led to better voice recognition systems. Enormous sets of tagged images have vastly improved automated image recognition(33).

Artificial intelligence advancements in a field follow the existence of Big Data in that field. A big impetus for the adoption of electronic health records in the United States was the passage of the Health Information Technology for Economic and Clinical Health (HITECH) Act, part of the American Recovery and Reinvestment Act of 2009. We are beginning to see the applications of machine learning to medicine now.

2.2 Diagnosis, prognosis in medicine vs. classification in machine learning

Diagnosis is a key part of medicine. A patient presents with several symptoms and laboratory tests. Based on these symptoms and tests, the physician is called on to classify the patient's condition into a disease. Once the diagnosis is complete, treatment can begin.

Classification, which will be discussed in the following section, is a key part of machine learning. A dataset is characterized by several variables. Based upon these variables, the machine learning algorithm is called on to classify the dataset into a particular class. An example is optical character recognition. An image containing a single numeral can be digitized into a 20x20 array of pixels. The algorithm can classify these 400-pixel values into one of ten possible digits, 0--9. Thus diagnosis in medicine is a natural target for the application of machine learning. Both medical diagnoses and machine learning classification share a set of terminology that describes the performance of diagnosis and classification tests. Diagnosis or classification tests are judged by their ability to correctly predict both the correct answer and avoid the incorrect answer. For the simplest case, a binary classification, four rates are relevant: the true positive, false positive, true negative and false negative rates. The true positive and true negative rates are the correct answers, while the false negative and false positive errors are the incorrect answers.

2.3 Sensitivity versus Specificity

These four numbers can be combined into two: sensitivity and specificity. The sensitivity is defined as the probability of a positive test given that the sample is known to be positive. While the specificity is the probability of a negative test given that the sample is known to be negative. Sensitivity and specificity depend on cutoff values. Different cutoff values give different sensitivity and specificity values. Ideally, a test will have both high sensitivity and high specificity, but a tradeoff exists between the two. Often a test will produce a numerical value. All values above a threshold have a positive test result, and all values below are negative. The exact values of the sensitivity and specificity will depend on the threshold value. If one chooses a low threshold value, one gets a high true positive rate (high sensitivity), and a high false positive rate (low specificity). On the other hand, if one chooses a high threshold value, one gets a low true positive rate (low sensitivity) and a low false positive rate (high specificity).

Characterizing such a test is a very general problem first addressed in the field of signal detection theory(34). The problem was posed this way: “Suppose an observer is given a voltage varying with time during a prescribed observation interval and is asked to decide whether the source is noise or is signal plus noise. What method should the observer use to make this decision and what receiver is a realization of that method?” Petersen, Birdsall and Fox answered the question they posed. The best method to decide whether you have a signal or just noise is to set a threshold. If the voltage exceeds the threshold, then one can claim to have detected the signal. Of course, this alters the question to “how

does one set a threshold." One sets the threshold based on the acceptable true positive rate and false positive rate. In the electrical engineers' formulation of the problem, the "test" was an electronic receiver that detected the voltage. Thus, the main task was to characterize their receiver's operating condition. They formulated a graphical expression of their receiver's performance that they named the receiver operating characteristic (ROC) curve. In our case, a better name might be the test characteristic curve, but the ROC nomenclature is firmly embedded in science.

2.4 ROC curves and area under the curve to quantify the quality of test

The receiver operating characteristic curve expresses the full capabilities of a test. One often needs to answer, “How good is the test?” The naïve would expect an answer like “80% accurate.” Where the naïve might define accuracy as “probability of a positive test given that the sample is known to be positive,” which we have defined as sensitivity.

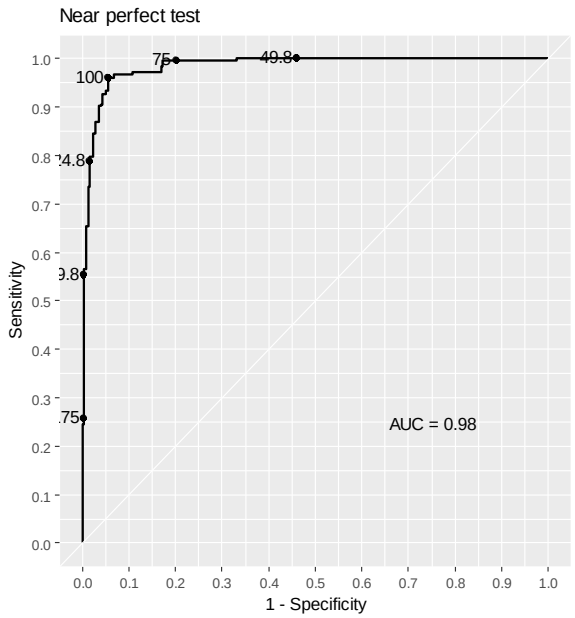

Figure 1. A near perfect test has an AUC close to 1.0. In this figure, the threshold values are indicated on the curve.

Instead, we can first define two extreme answers to the question, “how good is the test?” We can have the answer, “it is a perfect test,” and “it is a completely random test.”

A perfect test is one with 100% specificity and 100% sensitivity. It would have a ROC curve that looks like Figure 1.

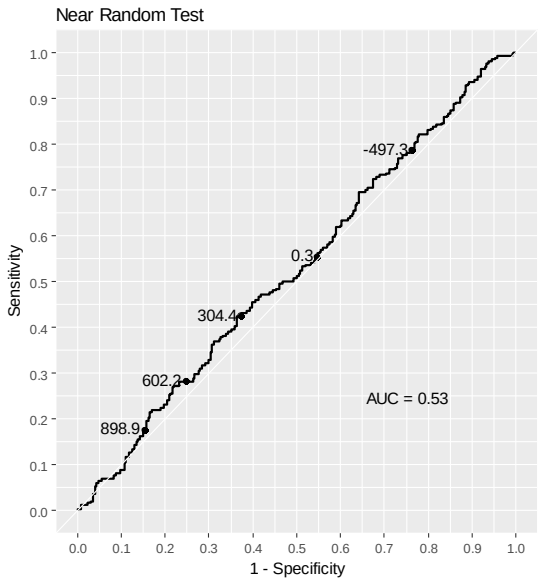

Figure 2. This ROC curve represents a poor test, one that is just a bit better than guessing at random.

If the predictions are made at random, the specificity will be equal to the sensitivity for all thresholds; it looks close to Figure 2.

The full range of possible threshold values and the associated true positive rate and false positive rate can only be expressed by a receiver operating characteristic curve, (ROC). However, a summary of the test's receiver operating characteristic curve can be computed by taking the integral under the receiver operating characteristic curve. This quantity is widely known as the area under the receiver operating characteristic curve or area under the curve or AUC. A perfect test has an AUC=1.0. A completely random test has an AUC=0.5. One can now answer the question, "how good is the test?" with a number. The test has an AUC=0.7. A useful shorthand is to think of the AUC as a grade one might get in a class.

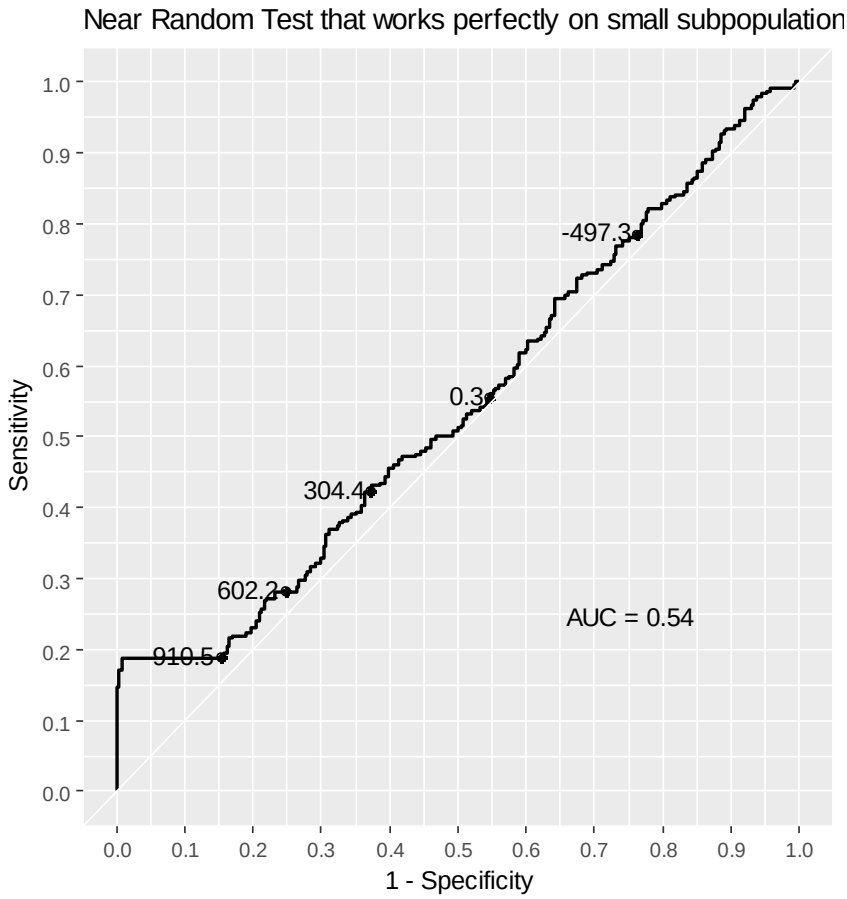

Figure 3 This test has almost the same AUC as the one in Figure 2, but it performs very well for a small subpopulation. This subpopulation performance is shown by the steep slope of the curve in the lower left. This figure demonstrates the importance of examine.

The AUC is only a summary of the test, and a test with a low AUC can be useful in a small subset of cases. Figure 3 shows an example from a BRCA-like test, where the test predicts outcome well for a small fraction of the population. The AUC is widely used to quantify tests in machine learning, medicine, psychology, and many other fields. In some fields, the AUC has other names including c-statistic, concordance statistic, and c-index. One appealing feature of the ROC / AUC is that it is insensitive to class imbalance. Suppose a test set contains 90% of normal patients and 10% diseased patients. The machine learning task is to classify whether a particular patient is normal or diseased. An algorithm that simply guesses “normal” for all unknown patients will have an accuracy of 90%. The AUC,

however, will be 0.50. The AUC is a better measure of the performance of the algorithm than the accuracy.

In cases of extreme class imbalance, screening for rare cancer, for instance, one often wants to identify the small number of patients most likely to be diagnosed with rare cancer. In these cases, it is better to practice to use a lift chart, which identifies what percentage of the target population (those with rare cancer) can be identified with the smallest possible group. As an example, suppose rare cancer occurs at a rate of 1 in 100,000. If we had an algorithm that could narrow identify a subset of the population in which rare cancer occurs at a higher rate, say 1 in 10,000, that algorithm would have a lift of 10. The lift is computed as the ratio of the rate after prediction to the rate before prediction. An algorithm that provides no information (random) has a lift of 1.

Table 2 Several common medical tests and their published AUC values.

Test	AUC	Reference
PSA to detect prostate cancer	0.68	(35)
Cardiac troponin to diagnose acute myocardial infarction	0.96	(36)
Cell-free DNA test for Down's syndrome	0.999	(37)
HbA1c for diagnosing diabetes	0.958	(38)
HEART score to predict major adverse cardiac events in 6 weeks from patients presenting with chest pain	0.86	(39)
Circulating tumor cells to diagnose lung cancer	0.6	(40)

2.5 Introduction of basic machine learning concepts

2.5.1 Supervised learning

The goal of supervised algorithms is to predict either a classification or a numerical result, called regression. Classification is ubiquitous in medicine. The patient presents with symptoms; the physician attempts to classify the patient, based upon symptoms into one of several possible diagnoses. Machine learning algorithms usually provide a measure of the probability that a set of data belongs to one class or another. Regression is less common but still useful. Regression can answer questions like, how many days is the patient expected to be in the hospital based on a set of data like age, initial diagnosis, vital signs, height, weight, days since last hospital stay, etc. There are many supervised machine learning algorithms, such as decision tress, neural network, and similarity learning.

2.5.2 Unsupervised learning

The word unsupervised means a specific label is not available before training. An unsupervised learning algorithm could learn to cluster samples automatically. The most common unsupervised machine learning algorithm is known as k-means clustering. The algorithm is rather old; it dates from at least the 1980's(41). The goal of this algorithm is to identify subgroups in the dataset. The number of subgroups must be pre-specified and is known as k. With a complex dataset, the algorithm will find a close to the ideal

partition of the data into k different clusters. Usually, subgroups have some important characteristic that makes them useful with a common diagnosis into subgroups. A typical process is to divide a group of patients into sub-groups, where each sub-group has a different survival time for that disease.

2.5.3 The basic process in machine learning

A machine learning algorithm takes a set of data to build a mathematical model to learn the patterns behind the data. From the training process, we determine the parameters in the model and find the parameters with the best performance. The data is called a training dataset. In the training dataset, each training sample is in the form of an array or a vector. The vector is composed of the variables that might affect the prediction. These variables are called features. So the vector is usually called a feature vector. The hyperparameters are the properties of the model that are determined before the determination of the weights of the individual neurons. Hyperparameters include the structure of the model (such as how many hidden layers or how many neurons in each layer in a neural network) and information on how the weights are to be determined during the training process, (factors are known as the learning rate, batch size, and momentum are examples.) Therefore, another dataset called the cross-validation dataset is used to get the hyperparameters with the best performance. Also, a dataset which is not seen by the model is used to evaluate the performance of a model. This dataset is called the test dataset. In a word, the model goes through training dataset, adjust the

hyperparameters on cross-validation dataset, and test on the test dataset to evaluate the performance. Figure 4 shows a basic machine learning procedure used in the following cancer prediction.

Supervised learning with hyperparameters requires that one be very careful to not overfit the data. Ideally, one has access to an unlimited dataset or more data than one can easily handle. Often, however, only a limited set of data is available. Given a limited dataset,

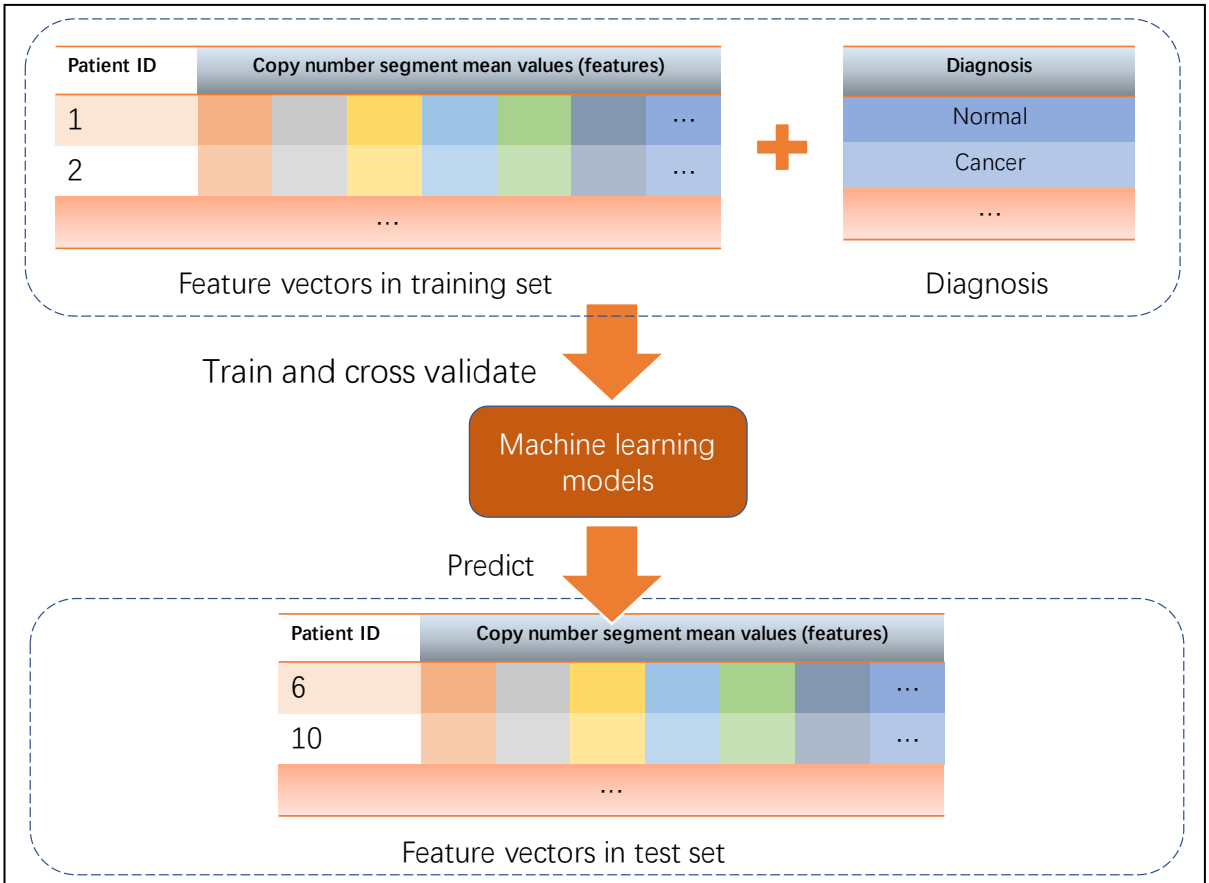

Figure 4 Basic machine learning procedure used in cancer prediction. Each block in the tables represents a feature. Each row in the tables represents the feature vector of each patient.

best practices are to split the data into three parts known as the training data, the validation data, and the test data. A typical split is 70% training, 15% validation, and 15% test data. One common pitfall is modifying the model after looking at how it performs

on the test data. Any such modification is a form of fitting to the test data. Improvements gained through such modifications inevitably will not translate to the next batch of fresh data. A second pitfall is when the data is preprocessed/sub-selected before the test data is extracted. This preprocessing can lead to information derived from the test data leaking into the training process.

The terminology for these sub-datasets: train, validate, the test is commonly used by the machine learning field, but it is not universal. Clinical laboratory tests are developed on one dataset and then validated on an independent dataset. Thus, clinical papers often refer to the final independent dataset as the validation step.

2.5.4 Overfitting

In many cases, the primary objective of a machine learning algorithm is to predict a value or diagnose a condition based upon the input. Biomedical examples abound:

- Can one diagnose whether a patient has lung cancer based upon an immunofluorescent studies of cells captured from the blood?

- Can one diagnose whether a patient has diabetes by measuring glycated hemoglobin levels in the blood?

- Can one predict whether a patient will have a heart attack in the next month based upon a combination of EKG, age, body mass index, tobacco smoking status, family history, and measurements of troponin in the blood?

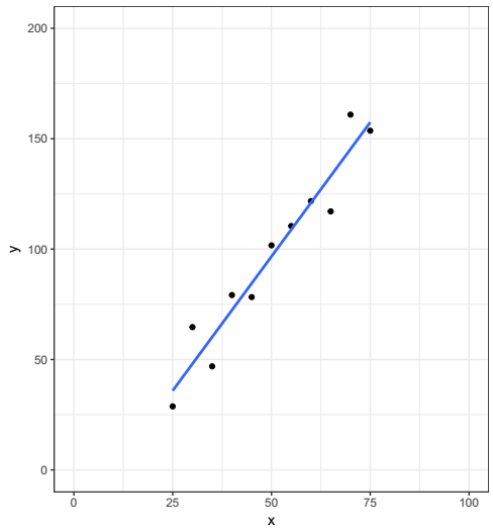

Figure 5. The line represents the best prediction of y, given x. The points indicate observations of y, and the corresponding x value. The line is represented by two parameters a slope and an intercept.

In the simplest case, one has an example set of results y and input values x. The goal is to identify the best function that will predict the value of y from the input values, x: $y = f(x)$, as shown in Figure 5. This process is referred to as supervised learning. The challenge is to derive this function f without overfitting. An example of overfitting is shown in Figure 6.

Every real set of data contains both signal and noise. The goal of machine learning is to build a model of the signal while ignoring the noise. The problem is that the noise is often indistinguishable from the signal.

Many approaches to reducing overfitting exist. The first, and simplest, is to have some fundamental understanding of the relationship between y and x. For instance, if there is a good reason to believe that y is linearly related to x, then the set of functions, f, should

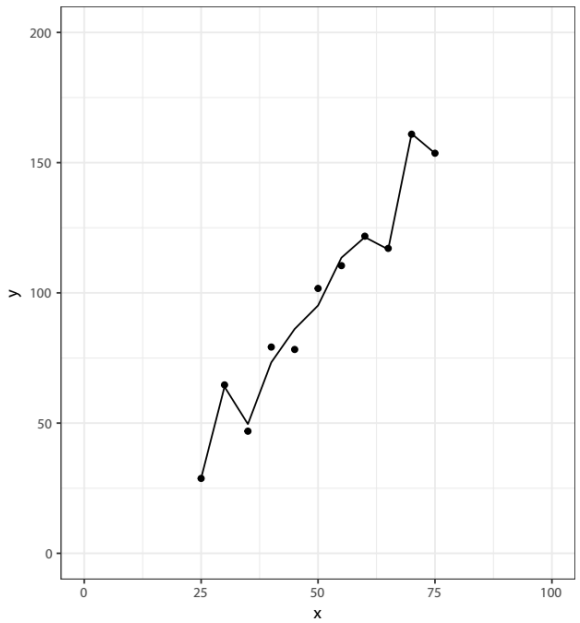

Figure 6. This black curve represents a clear case of overfitting. The same data as shown in Figure 4 are presented. In this case, the datapoints are fit with an 8th degree polynomial.

be limited to those in which x and y are linearly related: y= βx+c. If the relationship between y and x is complex and not well understood, then more complex methods are needed.

The main approaches to reduce overfitting in complex functions are known as regularization and drop out. To understand regularization better, we first need to state the cost function. A typical linear least square fit minimizes the cost function

$$C = \frac{1}{2n}\sum_{i=1}^{n}(y_i - \beta x_i)^2$$

This cost function is called the mean squared error. It takes the squared difference between the actual value, y_i and the predicted value β x. The problem can be posed as a minimization problem, where the goal is to minimize the cost function by adjusting β.

If multiple input variables, x_j, exist, instead of a single input variable, x, then a set of coefficients, βj is also needed. The cost function now looks like this, when there are a total of p input variables and n independent observations:

$$C = \frac{1}{2n}\sum_{i=1}^{n}\left(y_i - \sum_{j=1}^{p} x_{ij}\beta_j\right)^2$$

Minimizing this function will often lead to overfitting. Overfitting can be reduced by adding a new term to the cost function that penalizes functions that are more complex. This addition to the cost function is known as regularization. The idea is that a simpler model is one that uses fewer of the xj variables. The penalty is implemented by adding a term to the cost function that is proportional to the absolute value of the coefficient:

$$C = \frac{1}{2n}\sum_{i=1}^{n}\left(y_i - \sum_{j=1}^{p} x_{ij}\beta_j\right)^2 + \lambda\sum_{j=1}^{p}|\beta_j|$$

This addition to the cost equation is known as L1 or Lasso Regression. If, instead, the cost function includes a factor proportional to the square of the coefficient:

$$C = \frac{1}{2n}\sum_{i=1}^{n}\left(y_i - \sum_{j=1}^{p} x_{ij}\beta_j\right)^2 + \lambda\sum_{j=1}^{p}\beta_j^2,$$

The process is known as L2 regularization.

In each case, the cost function is increased by the final term, which is proportional to the parameter λ.

2.5.5 Regularization Revolution

Dropout is a revolutionary method to prevent overfitting (42). Dropout has primarily been applied to deep learning algorithms, but similar techniques are also used in other algorithms. The principle underlying the dropout mechanism is that by randomly dropping different connections within a network during the training process, the network becomes more robust and less susceptible to overfitting. The advent of dropout led to an immediate improvement in the performance of most deep learning algorithms.

Machine learning algorithms fall into two categories: supervised and unsupervised. Supervised algorithms require a batch of training data: a set of chest x-ray images from patients with pneumonia, for instance. The supervised algorithm uses the training data to build a model that can be applied to similar data (a chest x-ray) with an unknown diagnosis. An unsupervised algorithm is applied to a set of data to discover sub-classifications. For instance, an analysis based on six variables from patients with adult-

onset diabetes found that these patients could be grouped into five different types. The different types had a different risk of complications(43).

2.5.6 Applications

Applications of machine learning to medical data have grown rapidly in the past few years. We can organize these applications with different types of medical data. Examples of different data structures found in medical data include tabular data, medical images, time series, and natural language. In this dissertation, we developed many machine learning algorithms which applied to colorectal cancer prediction research.

Chapter3 Classification systems and gradient boost machine in cancer prediction

3.1 Pattern recognition in prediction research

In cancer prediction research, we can acquire a large amount of genetic data from patients' germline DNA. The prediction object in this study is to find the pattern behind the data and estimate the probability of developing cancers in the future. The act of taking in raw data and taking action based on the "category" is called pattern recognition(44).

A cancer prediction project is to separate the people with a high risk of getting cancer from genetic data. The variations in the genetic data are typically mathematical in form. The overarching goal and approach in pattern classification are to hypothesize the probability of getting cancer, process the higher associated genetic data, eliminate the unrelated data, and choose the best model that fit the pattern best.

In previous research, people use P-value of a simple chi-squared test to recognize the pattern in Genome-Wide Association Studies (GWAS)(45). GWAS are one of the most significant research in cancer prediction. It is an approach that involves rapidly scanning markers across the complete sets of DNA, or genomes, of many people to find genetic variations associated with a particular disease (46). In the GWAS study, the associations between Single-nucleotide polymorphisms (SNPs) and primary human cancers were reported. They focused on the phenotypes for a particular disease type. The genetic data

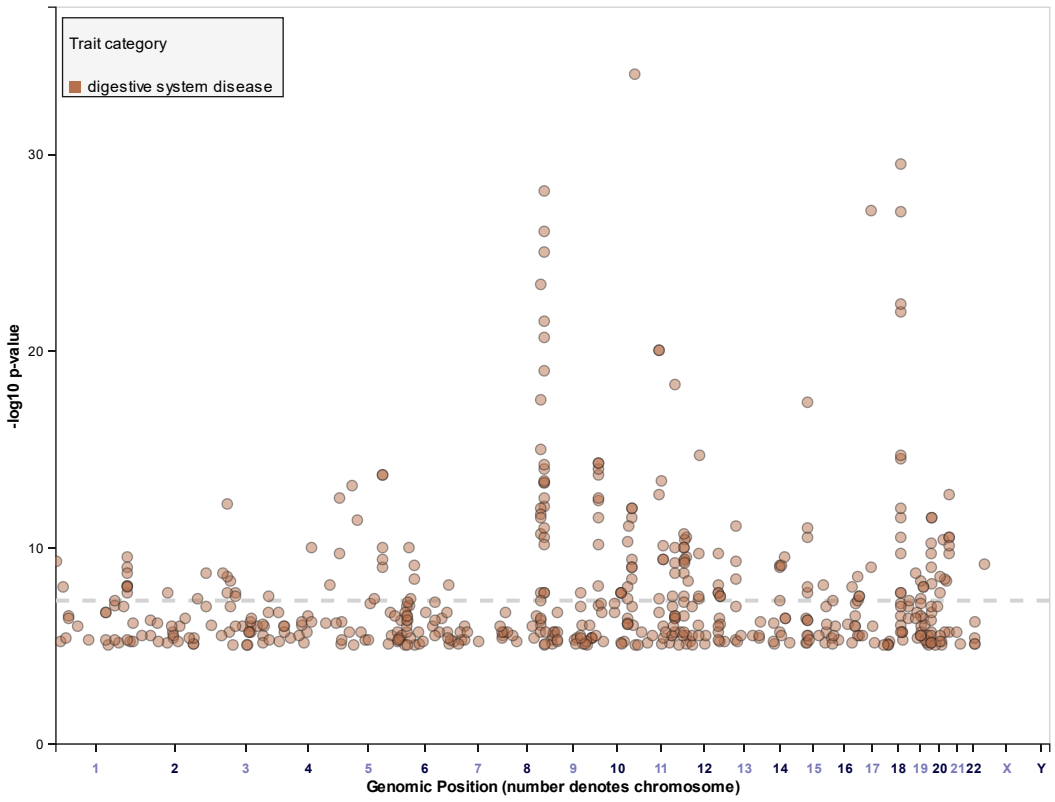

Figure 7 Manhattan plot of colorectal cancer in association with risk loci. The X-axis shows the genomic positions, and the Y-axis shows the P-values in association with digestive system disease (smaller P-value have a stronger association).

was acquired from the group of people with the disease (experimental group) and the group of people without the disease (control group). The genetic variants from the patients were then read by SNP arrays. The high frequency of a particular variant in the experimental group shows a higher association with the particular disease. Figure 7 shows a Manhattan plot of colorectal cancer in association with risk loci(47).

However, the GWAS study has limited success because it only focused on particular loci and neglected the inner connections between the loci. Copy number variation is a genetic variant that focuses on segments in the DNA sequence, which contains more information than single mutations (SNP). We used a copy of the (The Cancer Genome Atlas) TCGA data hosted by the Institute for System Biology Cancer Genomics Cloud, an interactive web-based application(48), to access and explore the rich cancer datasets.

Machine learning has the abilities to handle complex data, and it has broad application in bioinformatics. In this study, many machine learning techniques are applied to the research. Machine learning is a technique of computer science that uses statistical techniques to give computer systems the ability to progressively improve performance with the existing data, without being explicitly programmed(28). Computational statistics are intimately involved in machine learning, and it also focuses on prediction making using computational algorithms. The core idea of building a machine learning model is mathematical optimization. The efficiency of the optimization significantly affects the model quality.

3.2 Gradient Boost Machine

3.2.1 The principle of Gradient Boost Machine

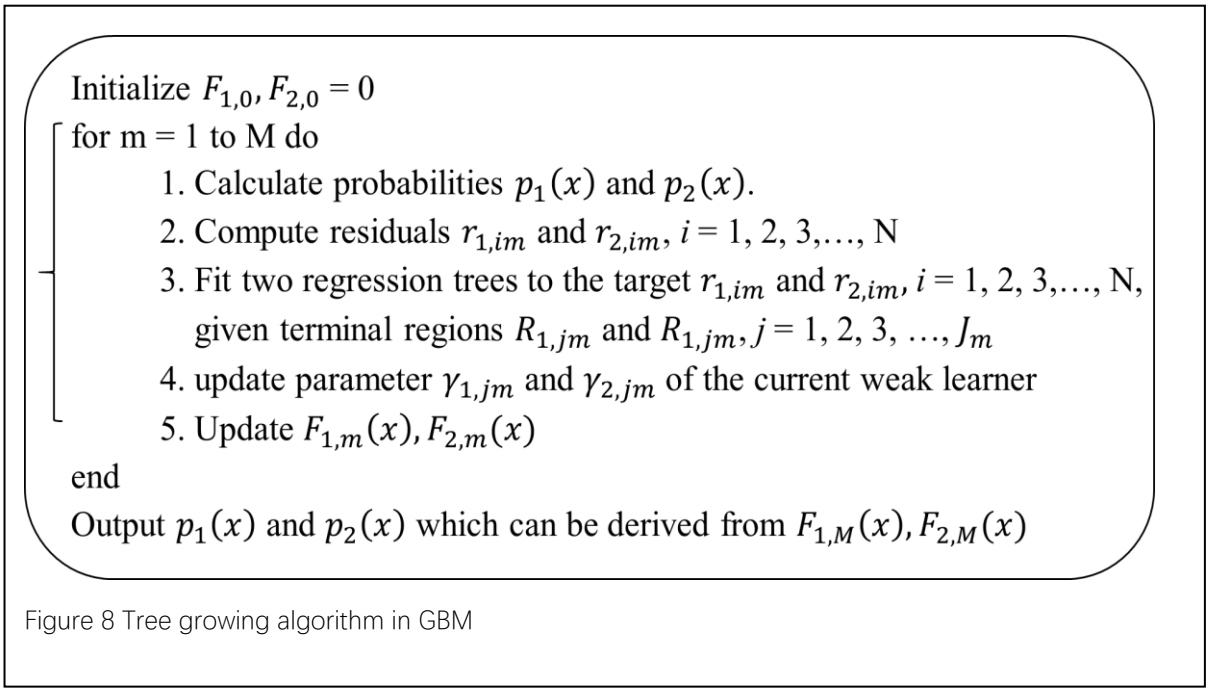

Figure 8 Tree growing algorithm in GBM

Gradient Boost Machine (GBM) was first implemented in this research. It first trains a group of weak learners and then ensemble them with different weights.

The GBM model grows following the algorithm specified by Hastie et al. (49) and is implemented by H2O.ai platform(50). A brief flow chart is shown in Figure 8. We trained a sequence of weak learners f_m in which regression trees were used as functions in a gradient boosting framework. This give a predicted score $\hat{y}$ by an estimate function $F(x)$, in the following form

$$F(x) = \sum_{m=1}^{M} f_m(x),\ f_m \in F$$

where x was the input feature vector, M was the number of regression trees, $f_m(x)$ were incremental functions ("steps" or "boosts") defined by an optimization method, and F was the space of all possible $f_m(x).B$ ecause our problem is a two-class classification problem, we built two estimate functions $F_1(x)$ and $F_2(x)$ to predict the score of colorectal cancer and non-colorectal cancer separately. We considered $F_1(x)\ and\ F_2(x)$ evaluated at each point x to be "parameters" and sought to minimize the loss function, which was given by

$$L(\{y_1, F_1\}, \{y_2, F_2\}) = -y_1 \log p_1(x) - y_2 \log p_2(x)$$

where $\mathrm{y}_1, \mathrm{y}_2 \in \{0,1\}$ were the observed values. $p_1(x)\ and\ p_2(x)$ were the probabilities when y_1 and y_2 were equal to one. It could be written as the following equation according to the symmetric multiple logistic transformation(51).

$$p_1(x) = exp\big(F_1(x)\big) / \Big(exp\big(F_2(x)\big) + exp\big(F_2(x)\big)\Big)$$

$$p_2(x) = exp\big(F_2(x)\big) / (exp(F_2(x)) + exp(F_2(x)))$$

The model was trained iteratively and additively. At the mth iteration, two new functions $f_{1,m}, f_{2,m}$(regression trees) selected from F are added to the ensemble $F_1(x)$ and $F_2(x)$ to predict the corresponding current residuals, $r_{1,im}$ and $r_{2,im}$, for the ith instance on the probability scale. The residuals could be computed as

$$r_{1,im} = y_{1,i} - p_{1,m-1}(x_i)$$

$$r_{1,im} = y_{2,i} - p_{2,m-1}(x_i)$$

Both of these trees have J-terminal nodes, splitting the data into corresponding regions $\{R_{1,jm}\}_{j=1}^{J}$ and $\{R_{2,jm}\}_{j=1}^{J}$. For big data splitting, this can be a problem. To avoid this problem, we approximate sorting with binning instead(52).

The parameters of f_m, $\gamma_{1,jm}$ and $\gamma_{2,jm}$ should optimize the mth objective function, which is given by

$$\gamma_{1,jm}, \gamma_{2,jm} = arg \min_{\{\gamma_{1,jm}, \gamma_{2,jm}\}} \sum_{i=1}^{N} (l(y_{1,i}, F_{1,m-1}(x_i) + \sum_{j=1}^{J} \gamma_{1,jm} I(x_i \in R_{1,jm})) + l(y_{2,i}, F_{2,m-1}(x_i) + \sum_{j=1}^{J} \gamma_{2,jm} I(x_i \in R_{2,jm})))$$

Where $l(y, F) = -ylogp$ from the loss function above, with F related to p through the probability equation above. N is the total number of patients in the training set, and the indicator function $I()$ has the value one if its argument is true, and zero otherwise. With a single Newton-Raphson step, the $\gamma_{1,jm}, \gamma_{2,jm}$ could be decomposed into a separate calculation for each terminal node of each tree(51),

$$\gamma_{k,jm} = \frac{\sum_{x_i \in R_{k,jm}} r_{k,im}}{2 \sum_{x_i \in R_{k,jm}} |r_{k,im}| (1 - |r_{k,im}|)} \quad k = 1, 2$$

At the end of each iteration, we can update the estimate function.

$$F_{k,m}(x) = F_{k,m-1}(x) + \sum_{j=1}^{J} \gamma_{k,jm} I(x_i \in R_{k,jm}), k = 1, 2$$

The final probability can be calculated by equation (3) and (4) with $F_{1,M}(x)$ and $F_{2,M}(x)$.

3.2.2 Feature engineering

Feature engineering focuses on feature extraction. The data made available by the ISB-CGC through Google Bigquery is organized into open-access datasets, which is made up of multiple tables. We used the datasets which are uniquely identified based on Google Cloud Platform(GCP) project name (ISB-CGC), and the datasets name (TCGA_hg38_data_v0 and TCGA_bioclin_v0). Each table inside the datasets is uniquely identified by the table name. The table, "Copy_Number_Segment_Masked," "Clinical," and "Biospecimen" were used in our study.

"Copy_Number_Segment_Masked" table, which is from "TCGA_hg38_data_v0" datasets, contains all available Copy Number Segmentation data across all TCGA samples. Each single copy number segment for each aliquot is shown in each row in the table. The field "chromosome" shows where each Copy Number Segment is located. The field "start_pos" and "end_pos" specify the coordinates for the segment, and "segment_mean" gives an estimate of the $log_2(Copy\ Numbner/2)$ mean value.

"Clinical" table, which is from "TCGA_bioclin_v0" datasets, contains over 11,000 cases. Each row in the table represents each TCGA case (patient or participant) with any clinical information.

"Biospecimen" table, which is also from "TCGA_bloclin_v0", is a sample-centric table. Each row in the table represents each of the TCGA samples. The filed "sample_type" specify

the sample type code. For example, in our study, we used Blood-Derived Normal sample (sample type code = 10) as our data source.

From TCGA datasets, the original datasets contain 69,518 copy number segments. We constructed our feature vector using three different methods: start grouping, end grouping, and no grouping. For start grouping, we incorporated the segments with the same start position as one feature. Then, we calculated the mean value of these copy number segment-mean values as a feature value. For end grouping, we incorporated the segments with the same end position as one feature. We then calculated the mean value of these copy number segment-mean values as a feature value. For no grouping, we used the original copy number segments as features. Finally, the gender vector was incorporated into the feature vector for each of the three methods. After model training, the start grouping method had the best performance.

Our data acquisition was performed using standard SQL. The program can be described in the following steps.

1. Selected the Copy Number Variation data from "Copy_Number_Segment_Masked" table and "Biospecimen" table with the same "sample barcode," in which the sample type is Blood-Derived Normal Sample.

2. Grouped the data to one row if it had the same start position at the same chromosome (start grouping) and counted the frequency of each location.

3. Sorted the data by order of frequency of the location. We select the top 30 rows from the table we got since most of the samples do not have the location information after the 30th row. The output data after this step is shown in Table 1.

4. From the “Copy_Number_Segment_Masked” table and “Clinical” table, selected the cancer data, gender data, and location data whose location information is available in Table 1.

5. Averaged the segment mean values if it is in the same row in Table 3.

6. Merged chromosome and start position as one unit in the table. Then spread the table to generate a feature vector in which each row represents a patient’s information (gender, age, segment mean values of different Copy Number Segment). The format of a typical feature vector is shown in Table 4.

Table 3 Frequencies of copy number segments in different start position

Row	Chromosome	Start position	Frequency	Row	Chromosome	Start position	Frequency
1	6	1011760	8861	16	11	456012	8852
2	18	326691	8861	17	1	3301765	8844
3	13	18874255	8861	18	15	23437561	8838
4	3	2170634	8861	19	X	3236359	8821
5	14	20033191	8861	20	2	480597	8821
6	17	1074619	8861	21	16	603333	8821
7	10	366509	8861	22	7	664936	8565
8	12	780472	8861	23	5	913983	7221
9	19	283868	8861	24	5	914118	1640
10	22	16934932	8861	25	19	30799692	376
11	4	1059384	8861	26	19	30797211	371
12	21	13974127	8861	27	13	106732887	365
13	9	789794	8860	28	13	106731890	360
14	20	472817	8860	29	21	24216643	348
15	8	667625	8859	30	7	24000259	348

Using feature vectors as an input, the GBM model produced the probability score of getting colorectal cancer. The feature vectors were divided into two parts, a training-cross validation set and a test set at the ratio of 0.85 to 1. The model was fitted by the training-cross validation set, and the results were given by the test set. The model hyper-

parameter setting includes the number of trees M, maximum tree depth P, and the number of folds for cross-validation K.

Table 4. A typical feature vector

feature	sample_barcode (not feature)	gender	1_3301765	10_366509
value	TCGA-2V-A95S-10D	MALE	9.00E-04	0.0011
feature	11_456012	12_780472	13_106731890	13_106732887
value	0	-0.002	NA	NA
feature	13_18874255	14_20033191	15_23437561	16_603333
value	0.001	-0.0013	0.0031	-6.00E-04
feature	17_1074619	18_326691	19_283868	19_30797211
value	0.0022	-0.001	0.0026	NA
feature	19_30799692	2_480597	20_472817	21_13974127
value	NA	0.006	0.0022	0.0018
feature	21_24216643	22_16934932	3_2170634	4_1059384
value	NA	0.0024	0.0024	0.0043
feature	5_913983	5_914118	6_1011760	7_24000259
value	0.0011	NA	5.00E-04	NA
feature	7_664936	8_667625	9_789794	X_3236359
value	0.0045	0.0028	-8.00E-04	-0.0028

Throughout implementing the model, we used P = 5, M = 100, K = 10 based on the default parameter set of gradient boosting training package according to H2O.ai.

3.2.3 Results for Gradient Boosting Machine

The performance of GBM model evaluation was implemented by ten-fold cross-validation on public colorectal cancer patient data with copy number variation information retrieved from ISB-GCC(48). We evaluated using receiver operating characteristic (ROC) curves(53), with the area under the ROC curve (AUC) as the evaluation criteria. As illustrated in Figure 9, the GBM model achieved an AUC of 0.8012, reflecting an effective prediction performance of the model.

As illustrated in Figure 10, all the copy number segment that we used as a feature vector is shown in the form of "XXX_XXX." The first part represents the chromosome where the copy number segment is located, and the second part represents the start position of the copy number segment. The importance of each copy number segment was also plotted showing that the copy number segment located on chromosome two has the largest effect on the probability of getting colorectal cancer.

A probability score of getting colorectal cancer was predicted for each patient in the test set, and all the patients were sorted at a descending order by the probability score. The patients who were diagnosed as colorectal cancer ("ground truth") and their probability score is shown in Table 5. The distribution of the patients who were diagnosed as colorectal cancer in the test set is shown in Figure 11. Setting a threshold of the probability score for the classification affects the true positive rate and false positive rate

significantly. So, we developed a new method to estimate the genomic predictability instead of using the accuracy of prediction in the previous study(54).

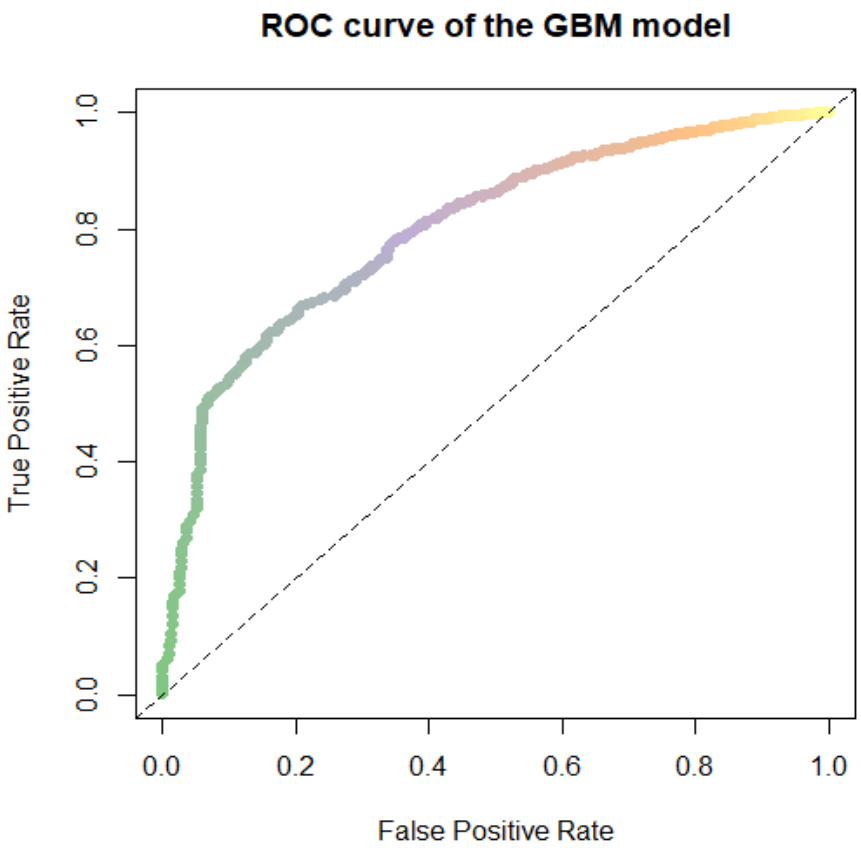

Figure 9 ROC curves of the GBM model. The AUC value of this ROC curve is 0.80, which indicates the GBM could efficiently predict the probability score

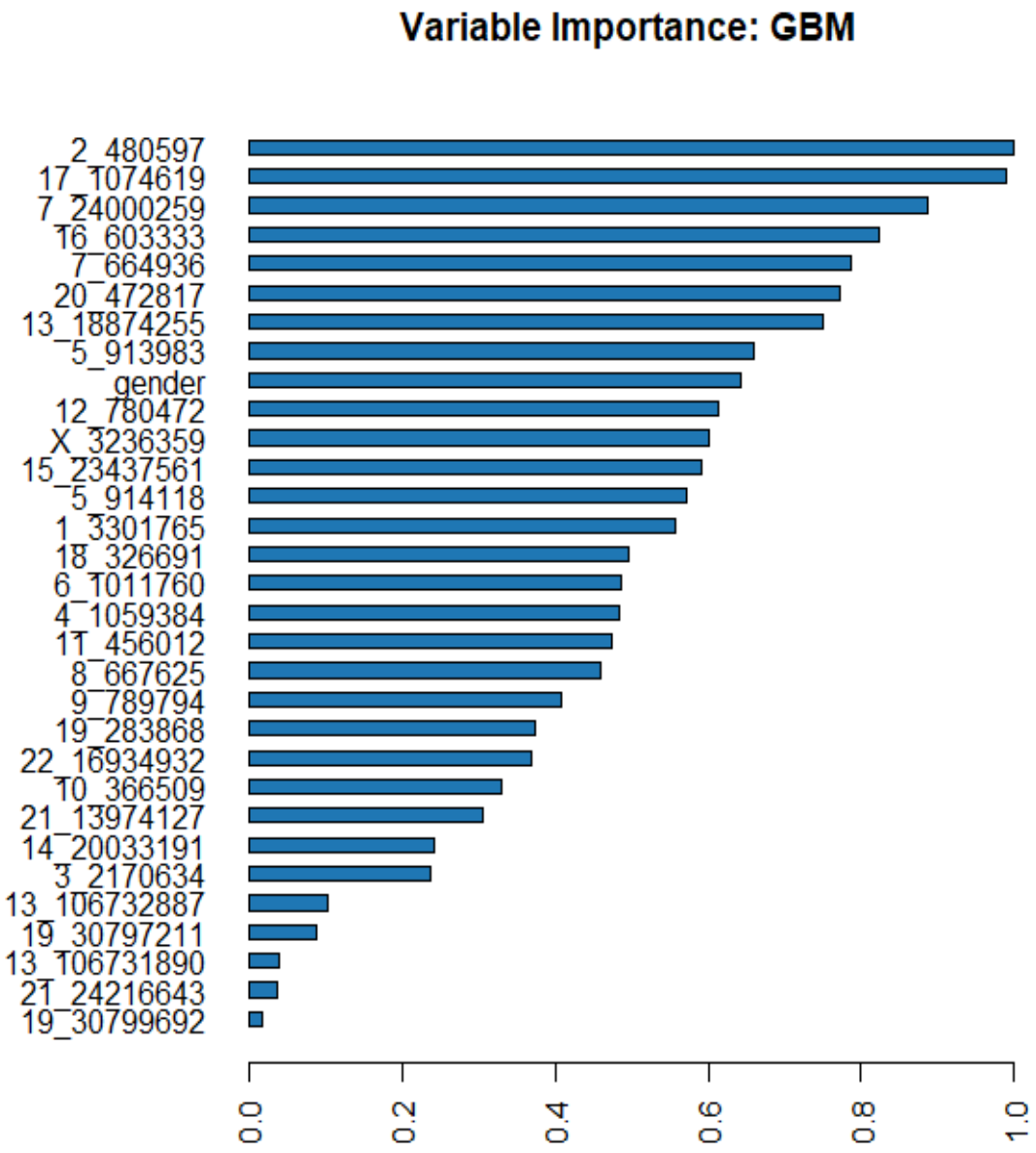

Figure 10 Copy Number Segment Importance. The Y-axis represents the copy number segment start position. In "XX_XXXX", the first number represents the chromosome where the copy number segment is located, and the second number represents the start position of the copy number segment. The X-axis represents the relative importance (the most important feature is set the value of one).

Table 5. Probability score of the diagnosed ("ground truth") colorectal cancer patients

Probability score	Serial number	Probability score	Serial number	Probability score	Serial number	Probability score	Serial number
0.884259	1	0.466803	8	0.460366	9	0.360891	16
0.337589	19	0.259559	23	0.244251	24	0.243751	25
0.231916	28	0.189347	40	0.168354	48	0.165799	49
0.164757	50	0.150394	63	0.146377	66	0.142975	71
0.13433	74	0.130431	76	0.128506	80	0.122948	84
0.11156	94	0.108889	99	0.095606	121	0.094155	127
0.092221	130	0.090953	131	0.085503	141	0.085178	143
0.080366	148	0.079359	149	0.076258	161	0.066001	192
0.060698	217	0.057163	234	0.053937	246	0.052	252
0.051889	254	0.051149	259	0.050043	264	0.0498	265
0.049429	269	0.049122	273	0.047636	282	0.041251	330
0.036157	361	0.034876	370	0.02853	439	0.028279	442
0.01969	543	0.015868	628	0.013698	689	0.011437	756
0.009948	809	0.00881	854	0.008163	887	0.007749	908
0.005998	1016	0.005842	1029	0.001515	1272		

The data above the blue line is the number of patients who are inherited from genome factor in level 1.
The data above the yellow line is the number of patients who are inherited from genome factor in level 4.
The data above the green line is the number of patients who are inherited from genome factor in level 6.
The data above the red line is the number of patients who are inherited from genome factor in level 8.

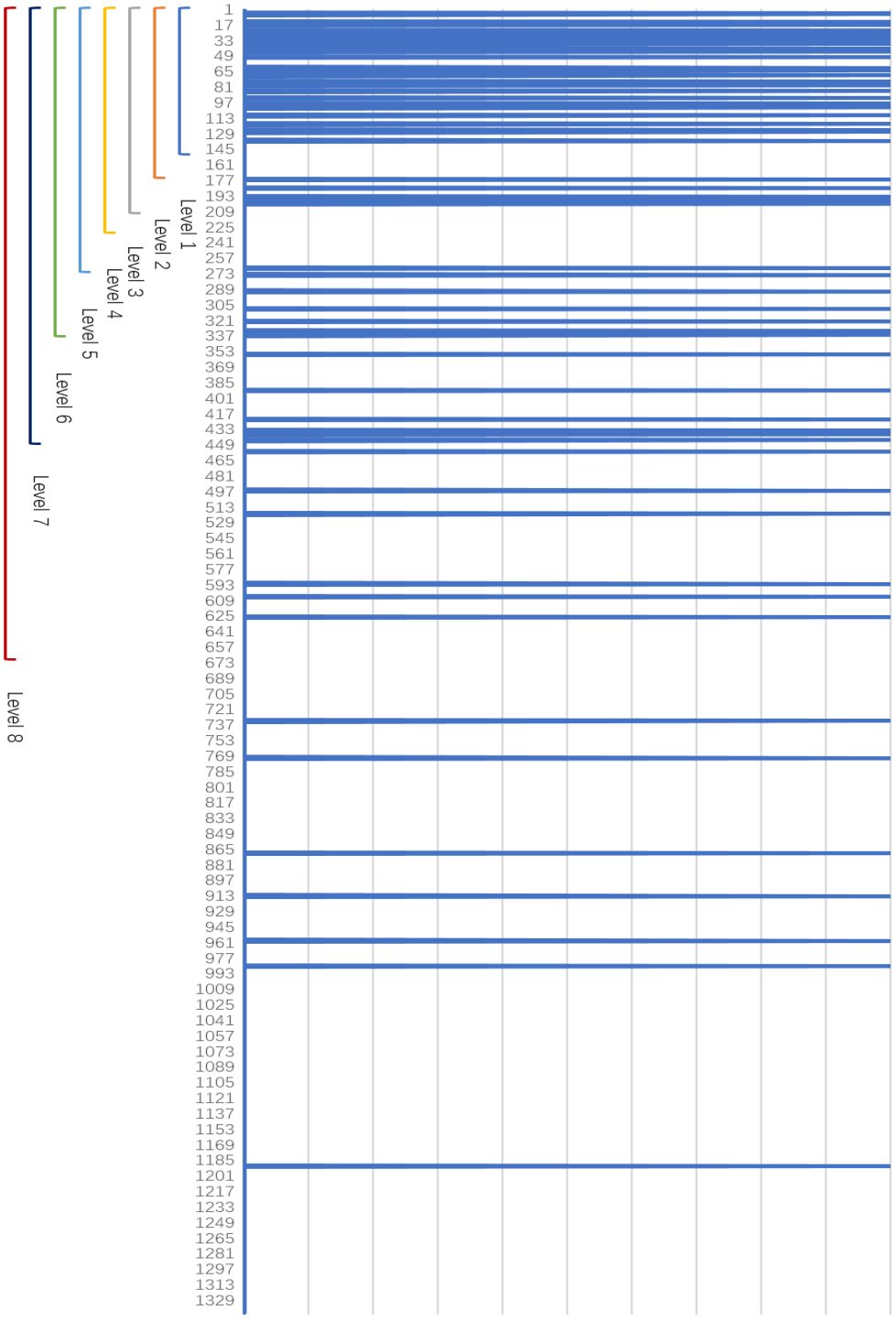

Figure 11 The distribution of the patients who were diagnosed as colorectal cancer in the test set. The patients in the test set was ranked by the probability score. Each patient is represented by a line (blue or white) in the figure. The blue line represents the patient who were diagnosed as colorectal cancer. The white line represents the patients who were not diagnosed as colorectal cancer.

3.3 Estimating the fraction of colorectal cancers which are predictable from germ line genetics

People usually estimated the heritability using the method in the literature on model fitting(55). Similar to the method, we estimated the predictability of colorectal cancer by the method below. After applying the GBM model to the test set data, we get a probability score for colorectal cancer in each patient. We then sorted the patient by probability in descending order. Each sample s in the test set was given a serial number j according to the order. The probability score of the diagnosed ("ground truth") colorectal cancer patients is shown in Table 6. According to how strictly the colorectal cancer is inherited from the genomic factor (Copy Number Variation), we measure the predictability in eight levels. The number of patients N_I who are inherited from genome factor in different level can be calculated by the following equation

$$N_I = N_C - (\frac{1}{k}) * \sum_{s \epsilon COAD} I(j > kN_T)$$

Table 6 k values used to set predictability level

level	1	2	3	4	5	6	7	8
k	8/9	7/8	6/7	5/6	4/5	3/4	2/3	1/2

Where N_C is the number of patients who were diagnosed as colorectal cancer patients, COAD is the set of patients who were diagnosed as colorectal cancer patients, and N_T is the total number of patients in the test set. k is a level parameter which varies by level. The specific value of k in different levels is shown in Table 4. Thus, the estimation of the predictability p can be derived from(56)

$$p = \frac{N_I}{N_C}$$

Figure 11 also represents how we estimated predictability at a different level. We estimated the predictability of colorectal cancer in eight different levels, which were ranked by how strictly the colorectal cancer was inherited from the genomic factor. The results are shown in Figure 12.

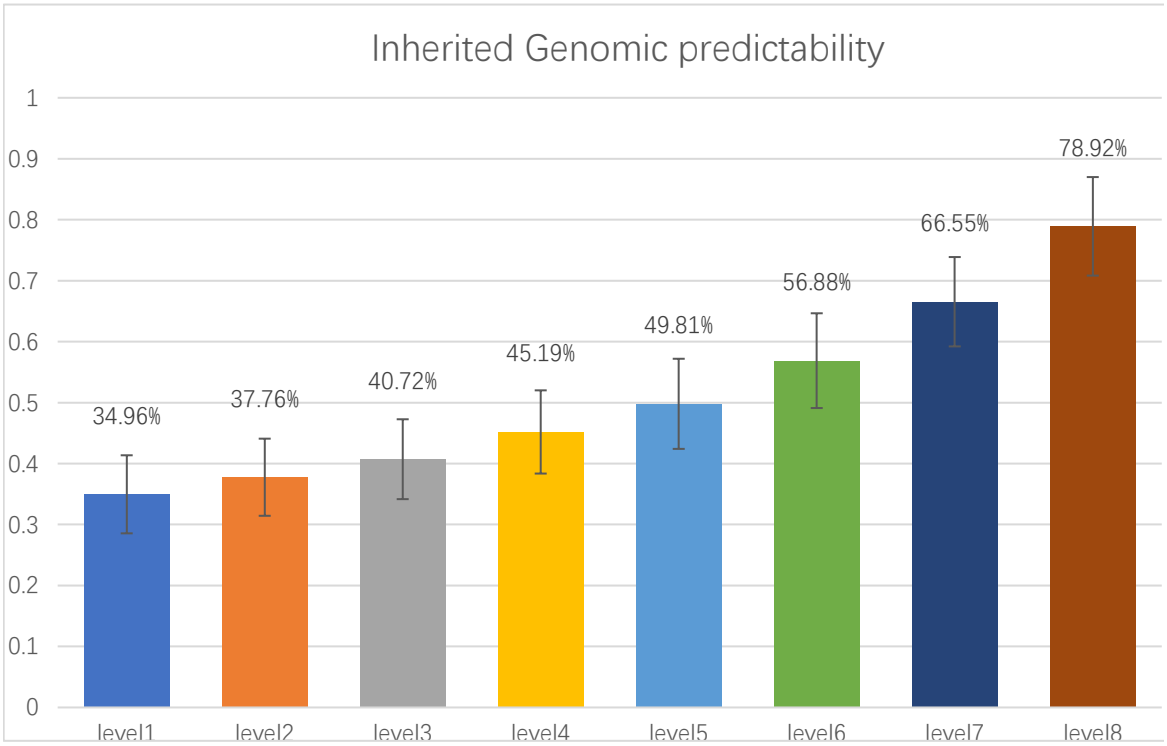

Figure 12 Inherited Genomic predictability in different level.

If we only consider whether the inherited genomic factor contributes to colorectal cancer, the predictability of colorectal cancer can be represented by the result of level 8. Thus, the inherited genomic predictability of colorectal cancer explained by copy number variation is estimated to be 78.92% $\pm$ 6.40%. On the other hand, heritability, which is defined as the proportion of phenotypic variation (V_P) that is due to variation in genetic values (V_G)(55), depend on the predictability significantly. However, the largest twin study

estimated that the heritability of colorectal cancer is 35%(26) and the heritability explained by all common SNPs is 7.42%(25). These studies shows that the SNPs explain only a small proportion of colorectal cancer heritability. Numerous studies have estimated the variance explained by the colorectal cancer SNPs(23)(19). However, the loci they identified only explained a small proportion of the heritability. As indicated by Jiao et al(25), a large part of the colorectal heritability is explained by other types of heritable factors, such as copy number variation. Our study strongly suggests that the copy number variation explains a certain portion of colorectal heritability, a factor has been ignored by the previous studies.

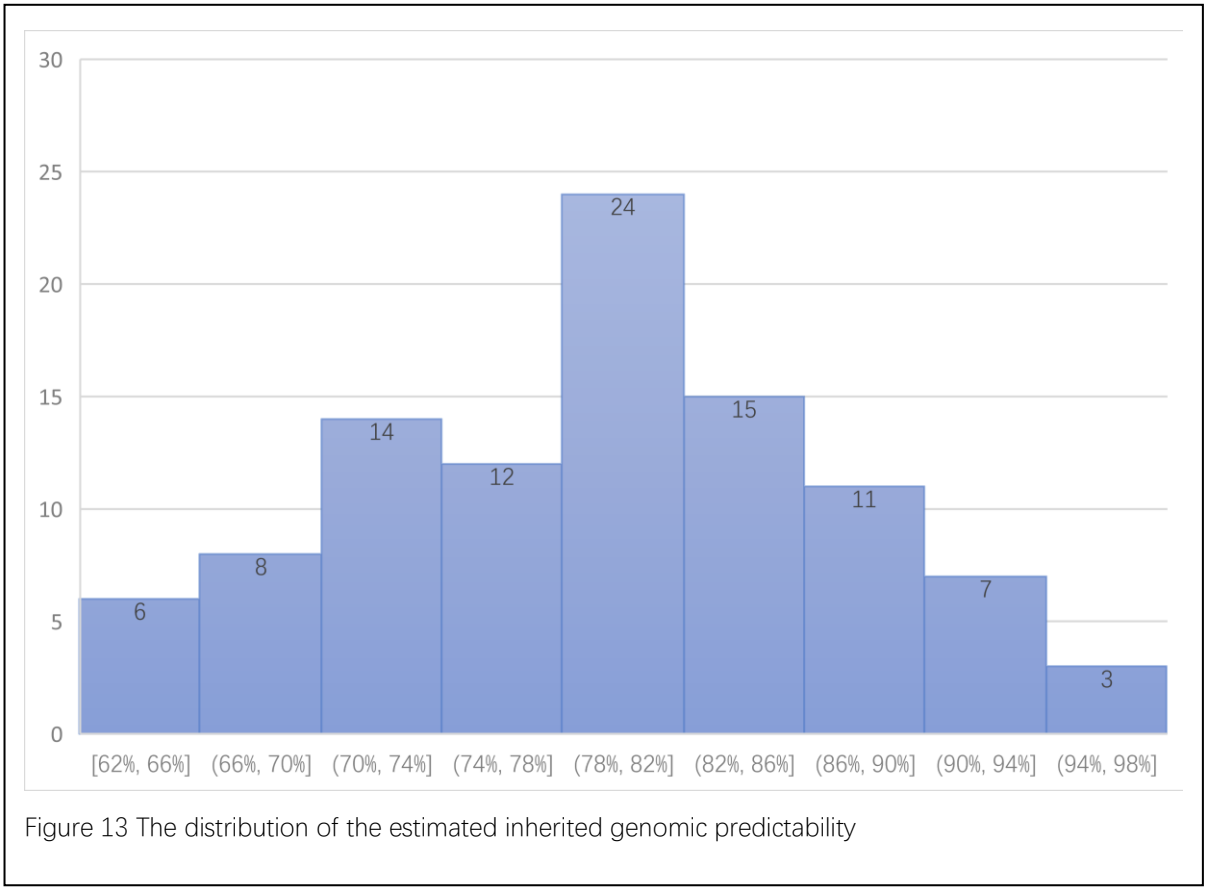

Figure 13 The distribution of the estimated inherited genomic predictability

To increase the precision of the heritability estimation, we built the GBM model 100 times on 100 randomly divided training-cross validation data. The distribution of the estimated inherited genomic predictability is shown in Figure 13. A Shapiro-Wilk normality test gives the p-value of 0.5251 which suggests that the predictability data follows a Gaussian distribution. Hence, a mean of these predictabilities was calculated to be the last estimated predictability.

Chapter4 Neuron network in cancer prediction

4.1 Artificial neural network

Like the human nervous system, the artificial neural network is a computing system composed of numerous neurons. The neural network itself is a complex computing algorithm. It can process up to thousands of data as multiple inputs. Such a system could be considered as a framework for many different machine learning algorithms to works together(57). The framework is not born to be programmed with any task-specific rules. It gradually learns from "training examples" to fulfill its functions. For example, In face recognition, the system will learn to identify a specific face by analyzing the images of human faces in many different angles. Before the neural network starts to learn, it has no prior knowledge about the face or any human face information, such as eyes, nose, mouth, or any facial details. Instead, It automatically gets this facial detail information during the training process.

The nervous system detects the environmental changes as input, then transmits signals to and from a different part of the body, finally generate an output response. Similarly, an artificial neural network generates an output response by analyzing the input values. Artificial neurons connect the output and input. Connections between artificial neurons are like the synapses in a biological brain. These connections are called edges. Each artificial neuron could process the signals from thousands of different neurons. Weights and bias in the edges would adjust to fit a particular pattern during training. Besides,

artificial neurons also have an activation function to filter the signal, such as setting up a threshold to only pass the signal crosses the threshold.

The Inputs of an artificial neural network aggregate a "layer," called the input layer. The neurons which are connected to the input layer aggregate the second layer. Similarly, the second layer can be followed by many layers until the output layer. These layers between the input layer and the output layer are called hidden layers. Different layers may perform different kinds of transformations on their inputs. Figure 14 shows a typical artificial neural network.

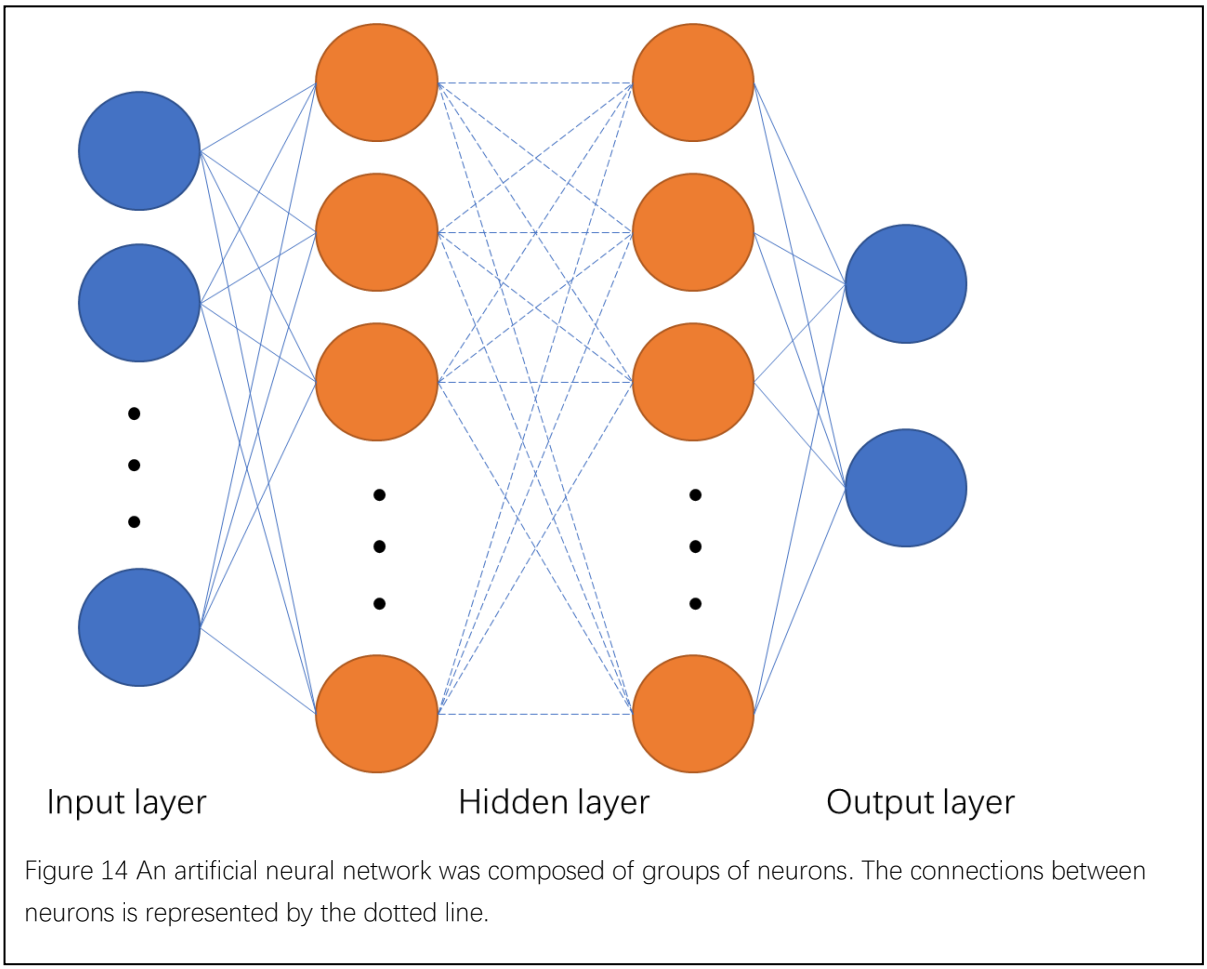

Figure 14 An artificial neural network was composed of groups of neurons. The connections between neurons is represented by the dotted line.

4.2 Neural network model

An artificial neuron works the way that a biological neuron model, but it is a mathematical model. A single input neuron could be explained in Figure 15(58). The input p is multiplied by the weight w and then add bias b to form the summer output $wp + b$. It then goes to an activation function f to finally produce the output a. In terms of comparison to the biological neuron, the weights explained by the analogy with the strength of synapse, the summer function and activation function explained by the

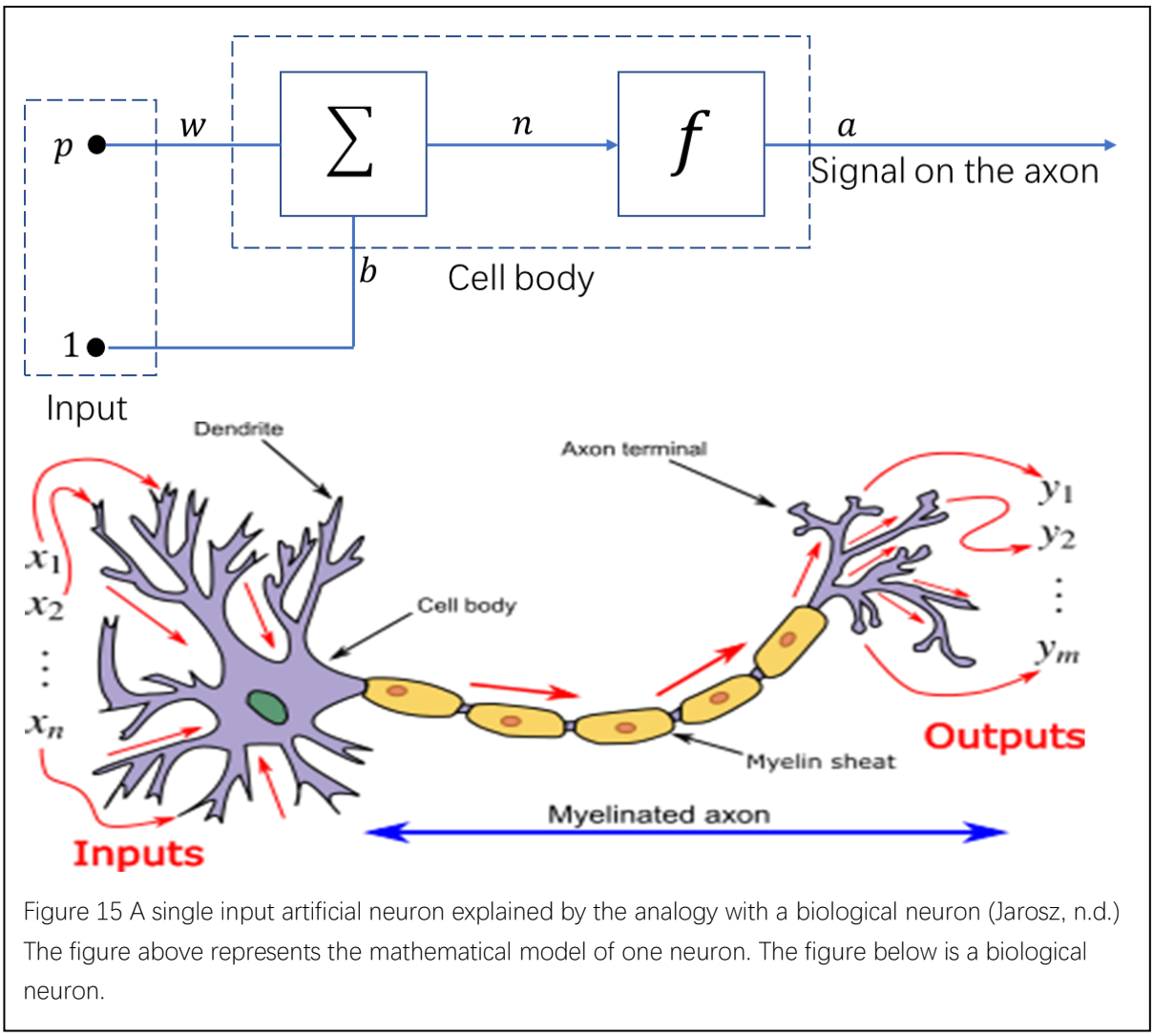

Figure 15 A single input artificial neuron explained by the analogy with a biological neuron (Jarosz, n.d.) The figure above represents the mathematical model of one neuron. The figure below is a biological neuron.

analogy with the cell body, and the output a explained by the analogy with the signal output on the axon.

Mathematically, the output of a single input neuron can be calculated by:

$$a = f(wp + b)$$

Here, the weight w and b are adjustable and will be tuned during the neural network training process. The initial values of w and b are set before network training. The initial values could also affect the training results. So, setting a proper initial value would be the first step in training a neural network.

Typically, the activation functions are usually nonlinear functions. Different activation function may apply to the different problems the artificial neural network is attempting to solve. The following are several common activation functions and their shape (Figure 16.).

- Logistic/sigmoid/soft step function:

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

- Rectified linear unit(ReLU)(59)

$$f(x) = \begin{cases} 0 \; for \; x < 0 \\ x \; for \; x \geq 0 \end{cases}$$

- Leaky rectified linear unit(Leaky ReLU)(60)

$$f(x) = \begin{cases} 0.01x \; for \; x < 0 \\ x \quad for \; x \geq 0 \end{cases}$$

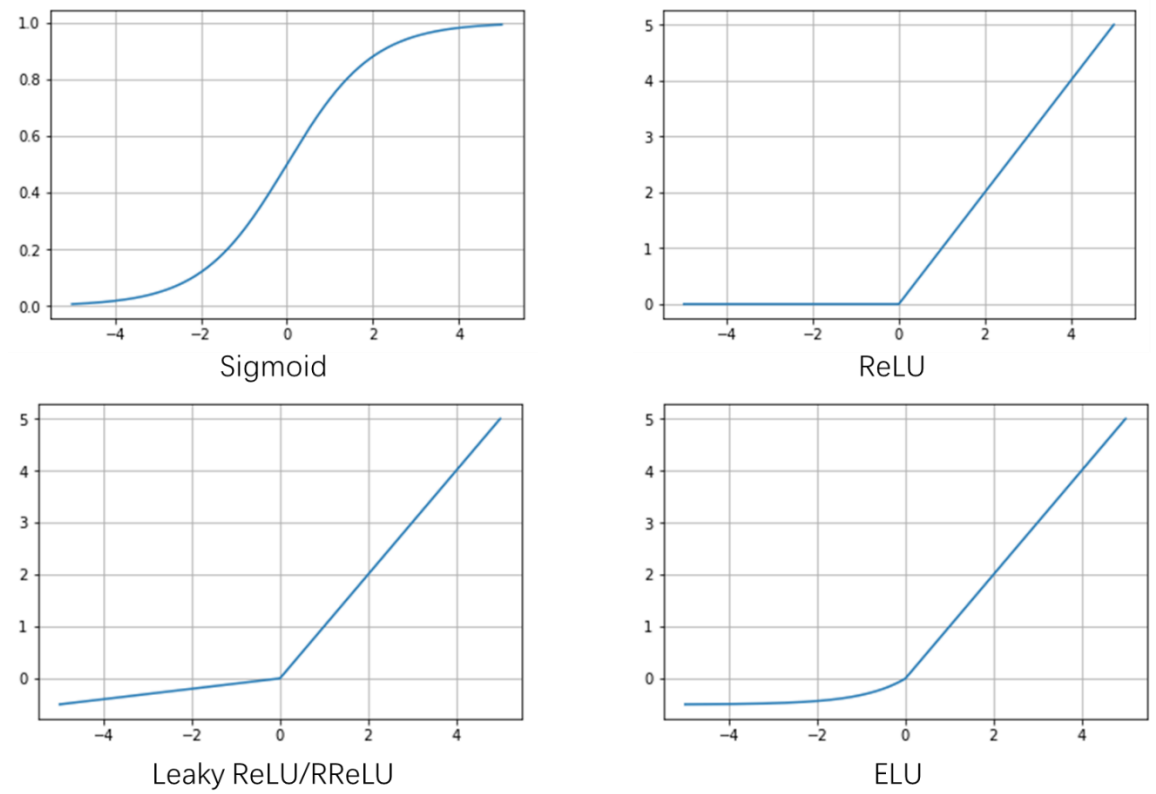

Figure 16 Common activation functions

- Randomized leaky rectified linear unit(RReLU)(61)

$$f(x) = \begin{cases} \alpha x \; for \; x < 0 \\ x \; for \; x \geq 0 \end{cases}$$

- Exponential linear unit(ELU)(62)

$$f(x) = \begin{cases} \alpha(e^x - 1) \; for \; x < 0 \\ x \qquad for \; x \geq 0 \end{cases}$$

Typically, an artificial neuron network has numerous inputs. A neuron with n inputs (represented by a one-dimensional matrix P ($(p_1, p_2, p_3, \ldots, p_n)$) (Figure 17.) is shown below.

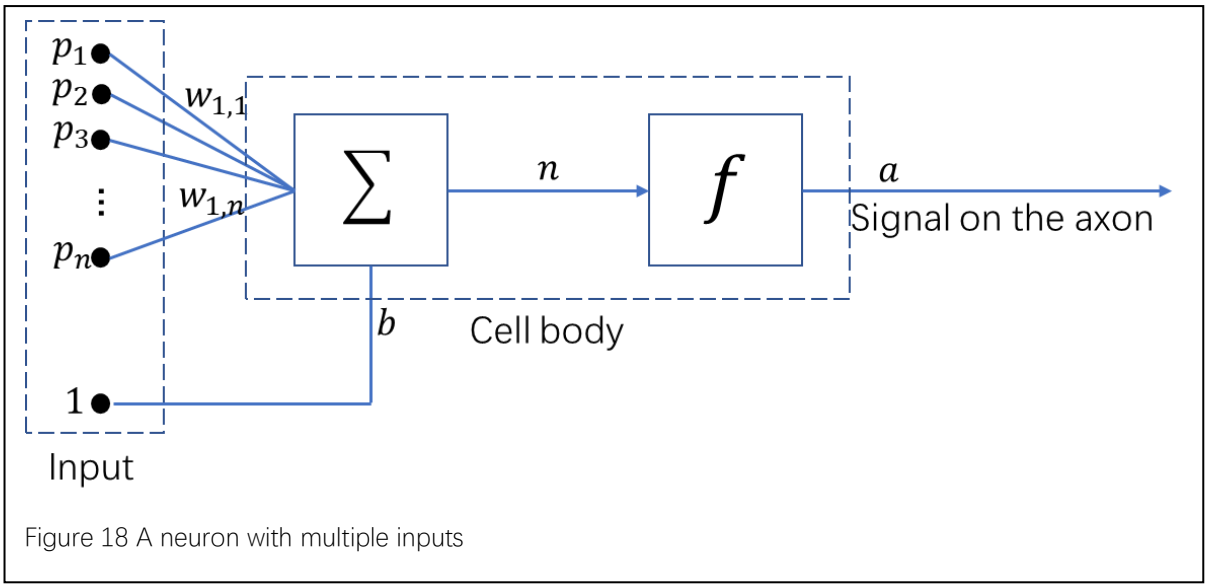

Figure 18 A neuron with multiple inputs

One-dimensional matrix W($w_{1,1}, w_{1,2}, w_{1,3}, \ldots, w_{1,n}$) and b represent the weights and bias for each input. Mathematically, the output a could be expressed as:

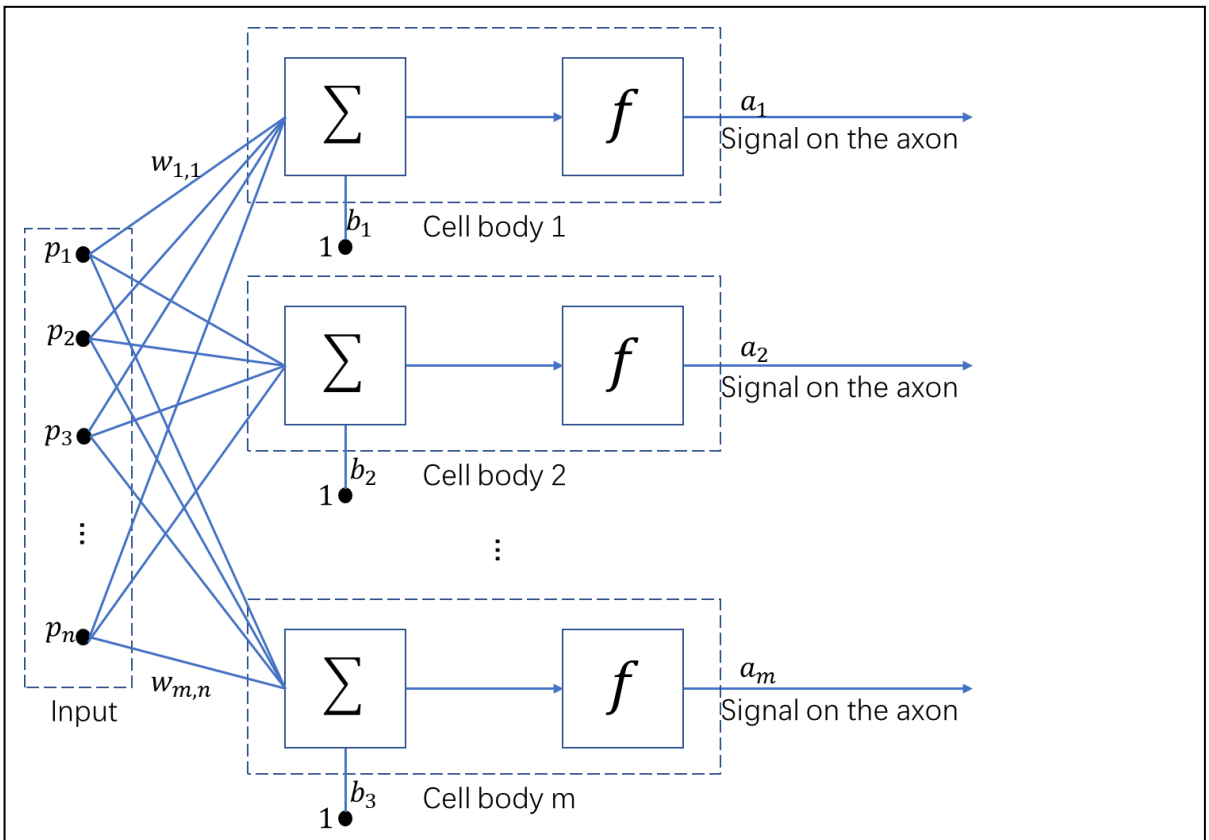

Figure 17 A layer of m neurons

$$a = f(WP' + b).$$

Typically, one neuron is not enough for complex computing. So, layers are aggregated by multiple neurons. Mathematically, the one-dimensional output A $(a_1, a_2, a_3, \ldots, a_m)$ from m neurons could be expressed as (Figure 18):

$$A = f(WP' + b)$$

However, most artificial neural networks are composed of many layers, especially the model used for deep learning. Several thousand layers may aggregate some deep learning architecture. For example, the state of the art image recognition model, ResNet, may have 1202 layers in its architecture(63). Figure 19 shows a two-layer network. If more layers are needed, additional layers should be added correspondingly. Again mathematically, it can be expressed as (the weights of the first layer is written as W^1, the

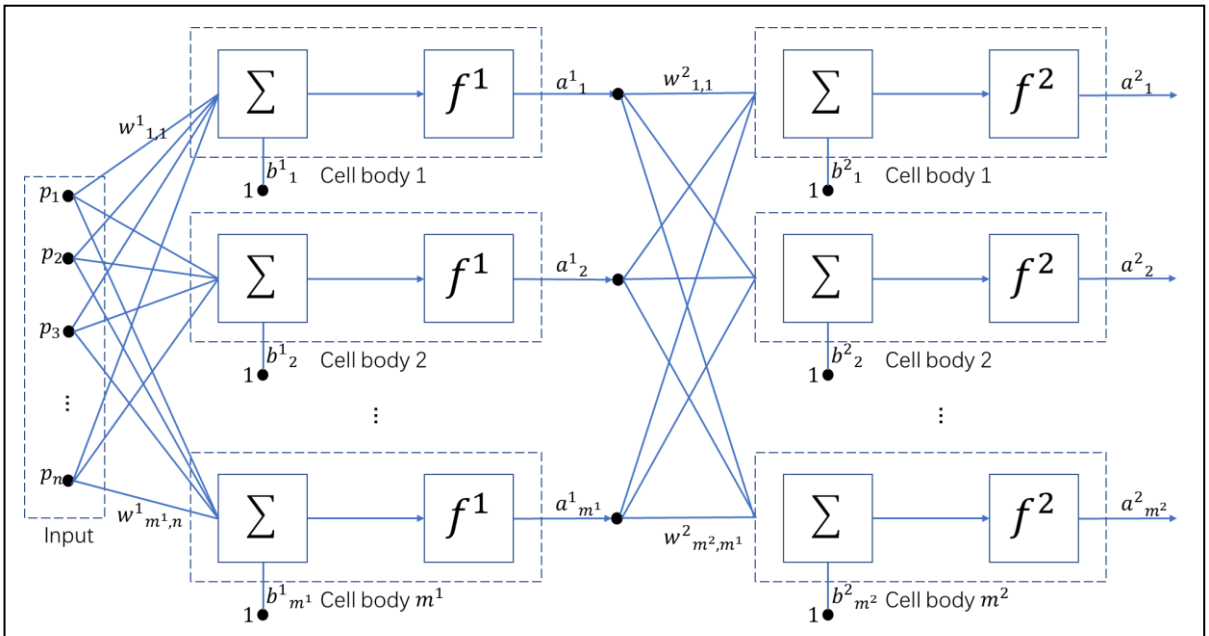

Figure 19 two-layer neural network

weights of the second layer is written as W^2, the bias of the first layer is written as b^1, the bias of the second layer is written as b^2, the activation function of the first layer is written as f^1, the activation function of the second layer is written as f^2)

$$a^2 = f^2(W^2 f^1(W^1 p + b^1) + b^2)$$

4.3 Artificial neural network learning

In cancer prediction, we mostly use an artificial neural network for supervised learning problems. For example, a neural network might be used for inferring the possibility of getting colorectal cancer. After seeing a lot of genetic data with a particular diagnosis for many patients, the artificial neural network needs to determine the probability of getting colorectal cancer accurately. So, in this chapter, we are about to focus on the learning process in supervised learning.

In most supervised learning problems, a particular pattern will be given a particular label to identify. The neural network needs to predict the label during the learning process, where the label must be a class label or a real number(64). In terms of cancer predicting problems, the patterns to be labeled are positive diagnosis and negative diagnosis. Usually, numerical number one is labeled to be the positive diagnosis and zero is labeled to be the negative diagnosis. The learning process for supervised learning problems has many significant advantages over a hard-wired system(64). The first is that it could save a significant amount of time for designing the hard-wired system. The second is that it could efficiently utilize the previous practical experiences through the learning process. Besides, it could be more precise for problems in which the environment often changes.

The cancer prediction problem is a typical binary classification problem, which means that the neural network needs to calculate the output either 0 or 1. As introduced in the previous section, a particular patient developed cancer from many complicated patterns. These patterns are unknown to us, at least we can not build a specific mathematic model

based on the potential cancer risks. The method by which the neural network adjust the parameters to fit the patterns constitutes the learning algorithms. So, the learning algorithms change the neural network data in response to the training data. After this learning process, we could assume that the model learns the patterns behind so many cancer risks.

The training data consist of paired data in the form of (x, y), where x is a specific patient's genetic copy number variation data, the input of the neural network ($x \in X$), and y is the diagnosis, the real fact ($y \in Y$). In classification problems, $Y = \{0, 1\}$. We assume that each patient's data is independent of others. The prediction of the neural network follows a probability distribution. The distribution reflects the probability that the patterns will be labelled in a particular way based on the cancer risk factors.

To elaborate on the training process mathematically and precisely, we assume that the model is a function of X and the output of the model is $f(x)$. A loss function $l(f(x), y)$, also called cost function, reflects the differences between the estimation $f(x)$ and the actual diagnosis y. Let F represents the set of functions $f_{w,b}(x)$ that parametrized by weight vector w and bias vector b. As descibed above, the objective of the tranining process is to find a model function f that could minimize the loss averaged on the training samples. In this study, we use gradient descent as the optimization method in the tranining process.

Gradient descent is an optimization algorithm that could efficiently find the local minimum of a function step by step. In each iteration, the algorithm takes a step proportional to the negative of the gradient towards the local minimum. Here, the function is the loss function, and the parameters (weights w and bias b) get updated each step. Mostly, gradient descent and its variations play a significant role in the machine learning process today. We can adjust the step size for the learning process to reach the local minimum as much as possible. The size of each step is called the learning rate. With a high learning rate, we can reach the local minimum faster but may cross over the minimum due to the big step we made. With a low learning rate, we can make sure every step is towards the local minimum since we update the gradient very frequently. Though a low learning rate could help each step moves precisely, it increases the cost of computing space and time.

Because we fix the input X in training and update the parameters each iteration, the loss function could be rewritten as a function of weights w and bias b. Moreover, in the most fundamental gradient descent method, we calculate the average loss for the whole training sample. Here we use a one layer neural network model to make the representation more clear (the derivation of the gradient for multiple layers will be discussed in backpropagation). So, we give the loss function in mean square error:

$$\hat{F}(w, b) = \frac{1}{N} \sum_{i=1}^{N} (y_i - (wx_i + b))^2$$

, Where N is the total number of patients in the training set. So, the gradient can be calculated as:

$$\hat{F}'(w,b) = \begin{bmatrix} \frac{\partial \hat{F}}{\partial w} \\ \frac{\partial \hat{F}}{\partial b} \end{bmatrix} = \begin{bmatrix} \frac{1}{N}\sum_{i=1}^{N} -2x_i(y_i - (mx_i + b)) \\ \frac{1}{N}\sum_{i=1}^{N} -2(y_i - (mx_i + b)) \end{bmatrix}$$

At the kth iteration, the weight w and b can be updated as:

$$w_{k+1} = w_k - \gamma \frac{\partial \hat{F}}{\partial w}$$

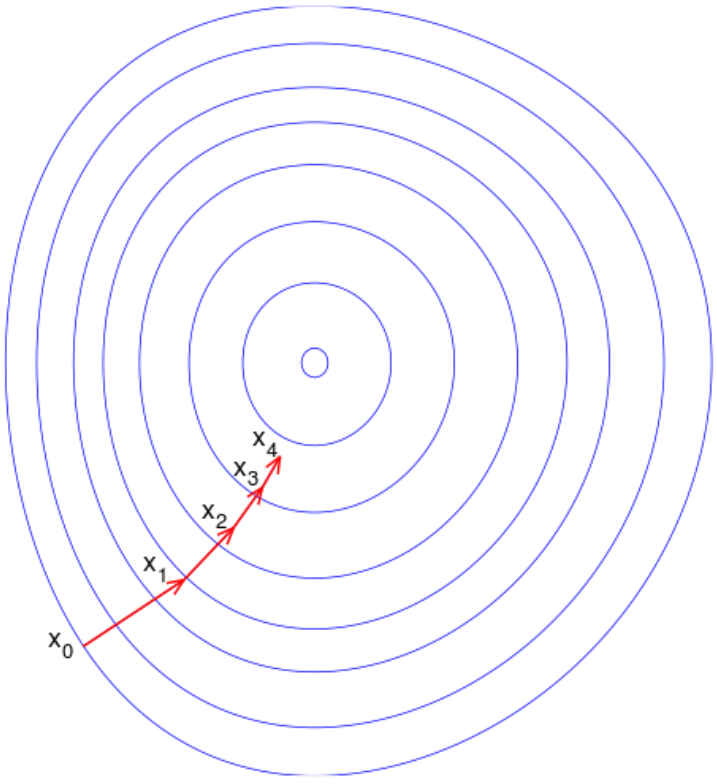

Figure 20 A brief illustration of gradient descent. The figure represents a loss function in a topographic map. The arrow towards the center represents the learning process to the local minima.

$$b_{k+1} = b_k - \gamma \frac{\partial \hat{F}}{\partial b}$$

, Where γ is the learning rate. We can adjust the total number of iteration and the learning rate until find the minimum most efficiently. When the function f is convex, the local minimum is also the global minimum. In this case, the algorithm also finds the parameters that converge to the global minimum. The learning process of a gradient descent algorithm could be illustrated in Figure 20(65).

Other variations of gradient descent were also applied to this study, such as mini-batch gradient descent, Stochastic gradient descent, gradient descent with momentum, Adam and Adagrad. The basic idea of these algorithms is similar. We applied as much as possible learning algorithms to make sure we get a more precise model.

4.4 Backpropagation

The gradient calculation is a significant step for neural network learning. From the last section, the gradient of a single layer neural network can be easily computed. However, the gradient calculation in multiple layer neural network with nonlinear activation function is more complicated.

Consider a multiple layer neural network shown in Figure 21. Still, applying the gradient descent of a mean square loss function for the mth layer weight and bias in the kth iteration:

$$w_{i,j}^{m}(k+1) = w_{i,j}^{m}(k) - \gamma \frac{\partial \hat{F}}{\partial w_{i,j}^{m}}$$

$$b_{i}^{m}(k+1) = b_{i}^{m}(k) - \gamma \frac{\partial \hat{F}}{\partial b_{i}^{m}}$$

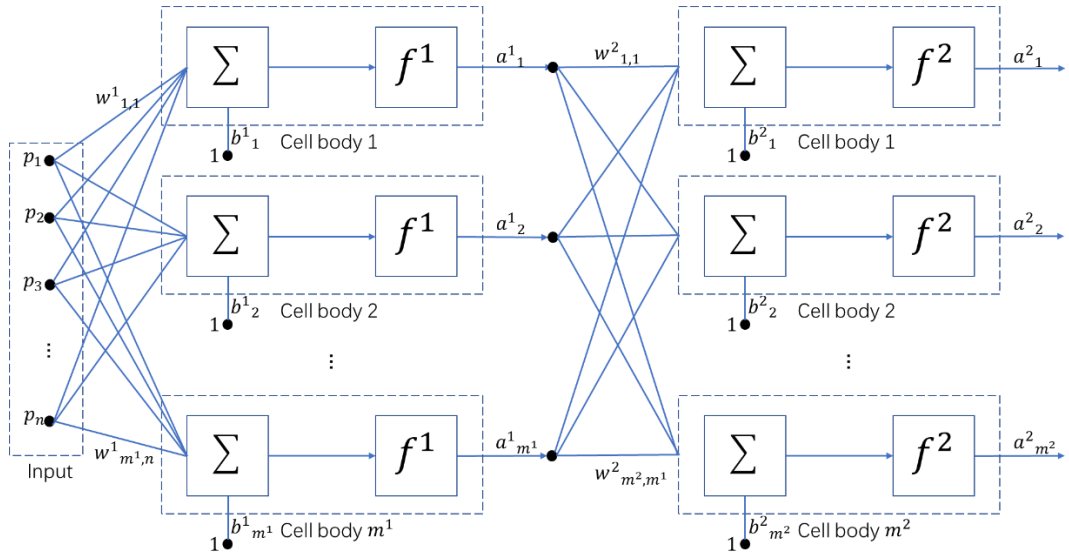

Figure 21 Multiple layer neural network

Here, the partial derivative of weights and bias in hidden layers are indirect to the loss function. So we will use the chain rule to calculate these derivatives(58).

$$a^2 = f^2(W^2 f^1(W^1 p + b^1) + b^2)$$

For a single neuron in the mth layer, we have

$$f^m(n_i^m) = f(\sum_{j=1}^{N^{m-1}} w_{i,j}^m a_j^{m-1} + b_i^m)$$

, Where N^{m-1} is the number of neurons of the last layer. $a_j^{m-1}$is the jth output from the last layer. Then we compute the derivative using the chain rule:

$$\frac{\partial \hat{F}}{\partial w_{i,j}^m} = \frac{\partial \hat{F}}{\partial n_i^m} \times \frac{\partial n_i^m}{\partial w_{i,j}^m}$$

$$\frac{\partial \hat{F}}{\partial b_i^m} = \frac{\partial \hat{F}}{\partial n_i^m} \times \frac{\partial n_i^m}{\partial b_i^m}$$

Therefore, we have the second term:

$$\frac{\partial n_i^m}{\partial w_{i,j}^m} = a_j^{m-1}$$

$$\frac{\partial n_i^m}{\partial b_i^m} = 1$$

To simplify the gradient equation, we define the sensitivity:

$$s_i^m \equiv \frac{\partial \hat{F}}{\partial n_i^m}$$

Written in matrix form,

$$\mathbf{s^m} \equiv \frac{\partial \hat{F}}{\partial \mathbf{n^m}} = \begin{bmatrix} \frac{\partial \hat{F}}{\partial n_1^m} \\ \frac{\partial \hat{F}}{\partial n_2^m} \\ \vdots \\ \frac{\partial \hat{F}}{\partial n_{N^m}^m} \end{bmatrix}$$

Then the gradient descent equation could be written as:

$$w_{i,j}^m(k+1) = w_{i,j}^m(k) - \gamma s_i^m a_j^{m-1}$$

$$b_i^m(k+1) = b_i^m(k) - \gamma s_i^m$$

Now, the calculation of s_i^m becomes the only thing left. The main idea of the process is that we compute the sensitivity of the mth layer from the sensitivity of the (m + 1)th layer, which explain the term backpropagation. From the chain rule, we have:

$$s_j^m \equiv \frac{\partial \hat{F}}{\partial n_j^m} = \frac{\partial n_i^{m+1}}{\partial n_j^m} \times \frac{\partial \hat{F}}{\partial n_i^{m+1}} = \frac{\partial n_i^{m+1}}{\partial n_j^m} \times s_j^{m+1}$$

Where,

$$\frac{\partial n_i^{m+1}}{\partial n_j^m} = \frac{\partial (\sum_{j=1}^{N^{m-1}} w_{i,l}^{m+1} a_l^m + b_i^{m+1})}{\partial n_j^m} = w_{i,j}^{m+1} \frac{\partial a_j^m}{\partial n_j^m}$$

$$= w_{i,j}^{m+1} \frac{\partial f^m(n_j^m)}{\partial n_j^m}$$

Then, apply the equation above to the chain rule:

$$s_j^m = w_{i,j}^{m+1} \frac{\partial f^m(n_j^m)}{\partial n_j^m} s_j^{m+1}$$

Written in matrix form, we have:

$$\frac{\partial \mathbf{n}^{m+1}}{\partial \mathbf{n}^m} \equiv \begin{bmatrix} \frac{\partial n_1^{m+1}}{\partial n_1^m} & \frac{\partial n_i^{m+1}}{\partial n_2^m} & \cdots & \frac{\partial n_1^{m+1}}{\partial n_{N^m}^m} \\ \frac{\partial n_2^{m+1}}{\partial n_1^m} & \frac{\partial n_2^{m+1}}{\partial n_2^m} & \cdots & \frac{\partial n_2^{m+1}}{\partial n_{N^m}^m} \\ \vdots & \vdots & & \vdots \\ \frac{\partial n_{N^{m+1}}^{m+1}}{\partial n_1^m} & \frac{\partial n_{N^{m+1}}^{m+1}}{\partial n_2^m} & \cdots & \frac{\partial n_{N^{m+1}}^{m+1}}{\partial n_{N^m}^m} \end{bmatrix}$$

$$= \mathbf{W}^{m+1} \dot{\mathbf{F}}^m(\mathbf{n}^m)$$

Where

$$\dot{\mathbf{F}}^m(\mathbf{n}^m) = \begin{bmatrix} \frac{\partial f^m(n_1^m)}{\partial n_1^m} & 0 & \cdots & 0 \\ 0 & \frac{\partial f^m(n_2^m)}{\partial n_2^m} & \cdots & 0 \\ \vdots & \vdots & & \vdots \\ 0 & 0 & \cdots & \frac{\partial f^m(n_{N^m}^m)}{\partial n_{N^m}^m} \end{bmatrix}$$

Finally, the sensitivity $\mathbf{s}^m$ can be written as:

$$\mathbf{s}^m = \frac{\partial \hat{F}}{\partial \mathbf{n}^m} = \left(\frac{\partial \mathbf{n}^{m+1}}{\partial \mathbf{n}^m}\right)^T \frac{\partial \hat{F}}{\partial \mathbf{n}^{m+1}} = \dot{\mathbf{F}}^m(\mathbf{n}^m)(\mathbf{W}^{m+1})^T \frac{\partial \hat{F}}{\partial \mathbf{n}^{m+1}}$$

$$= \dot{\mathbf{F}}^{m}(\mathbf{n}^{m})(\mathbf{W}^{m+1})^{T}\mathbf{s}^{\boldsymbol{m+1}}$$

From the equation above, the $\mathbf{s}^{\boldsymbol{m}}$ follows the backpropagation rule (from the last layer to the first layer). Now, we finish the calculation of the gradient of each parameter. Apply it to the gradient descent equation, one iteration of the neural network learning process is finished.

Chapter5 Wide model for colorectal cancer prediction

5.1 Wide model

The word “wide” in the wide model refers to a large number of inputs with a single layer neural network architecture. In past decades, there were so many researches on studying which model could do better on various problems. However, no particular model could always perform well in all the tasks. In recent years, deep neural networks have been shown state-of-the-art results for many problems. For example, Convolutional neural network performs well in image recognition problems; Recurrence neural network performs well in natural language processing. In these situations, networks with multiple layers could beat shallow networks easily(66). The outperformance of deep architectures makes it popular in machine learning literature. However, the algorithm for deep architecture often relies on the number of epochs, learning rate, batch size, etc. Tuning these parameters can be tricky in training and could affect the performance significantly.

Moreover, multiple layer networks have a better performance with the large training dataset. However, in cancer prediction research, it is hard to acquire large dataset because of the unavailability of enough patients. So, in this study, we tried many architectures and tested its performance.

For small scale classification problems, wide model (generalized linear model) are widely used because it is simple to be implemented and easy to interpret. So we first test the performance of wide model in this study.

5.2 Feature engineering

5.2.1 Dense feature and sparse feature

In mathematics, “dense” means non-zero numbers in a matrix and “sparse” means zero numbers in a matrix. If a matrix consists of mostly zeros and few non-zero numbers, we call it a sparse matrix. On the contrary, if a matrix consists of mostly non-zero numbers and few zeros, we call it a dense matrix. In this study, if the genetic data from a patient is missing, we replace it with zero in the matrix. So, the copy number variation data could be partially dense and partially sparse in the feature matrix because many segment-mean values for a particular segment was missing in the original dataset.

As the feature processed in Gradient Boost Machine, feature vectors were extracted by each patient. In machine learning, a large scale of sparse data could cause many problems. It is space consuming, time-consuming, and make neural network hard to learn at the beginning of the training process. So, we process the dense feature and dense feature separately in some architecture in this study. Because we do not do sparse feature embedding in wide architecture, an input vector which is concated by 25 dense features and 250 sparse features are used for wide model training. Table 7 shows a typical feature vector used for wide neural network training.

Table 7. A typical feature vector for the wide model

feature	sample_barcode (not feature)	gender	1_3301765	11_456012
value	TCGA-2V-A95S-10D	MALE	9.00E-04	0
feature	10_366509	12_780472	16_603333	13_106732887
value	0.0011	-0.002	-6.00E-04	0
feature	13_18874255	14_20033191	15_23437561	13_106731890
value	0.001	-0.0013	0.0031	0
feature	17_1074619	18_326691	19_283868	19_30797211
value	0.0022	-0.001	0.0026	0
feature	21_13974127	2_480597	20_472817	19_30799692
value	0.0018	0.006	0.0022	0
feature	4_1059384	22_16934932	3_2170634	21_24216643
value	0.0043	0.0024	0.0024	0
feature	5_913983	X_3236359	6_1011760	7_24000259
value	0.0011	-0.0028	5.00E-04	0
feature	7_664936	8_667625	9_789794	5_914118
value	0.0045	0.0028	-8.00E-04	0

- Features on the left side of the blue line are the dense feature
- Features on the right side of the blue line are the sparse feature.

5.3 Wide model architecture

The whole architecture was built based on Tensorflow framework(67). The architecture is built based on the following steps:

1. Build a function of embedding layers with weights and bias.

2. Build a function of fully connected layers with weights, bias, and activation function.

3. Set the number of neurons for the first layer, and distribute "placeholder" for the neurons

4. Connect the neurons with the embedding layer

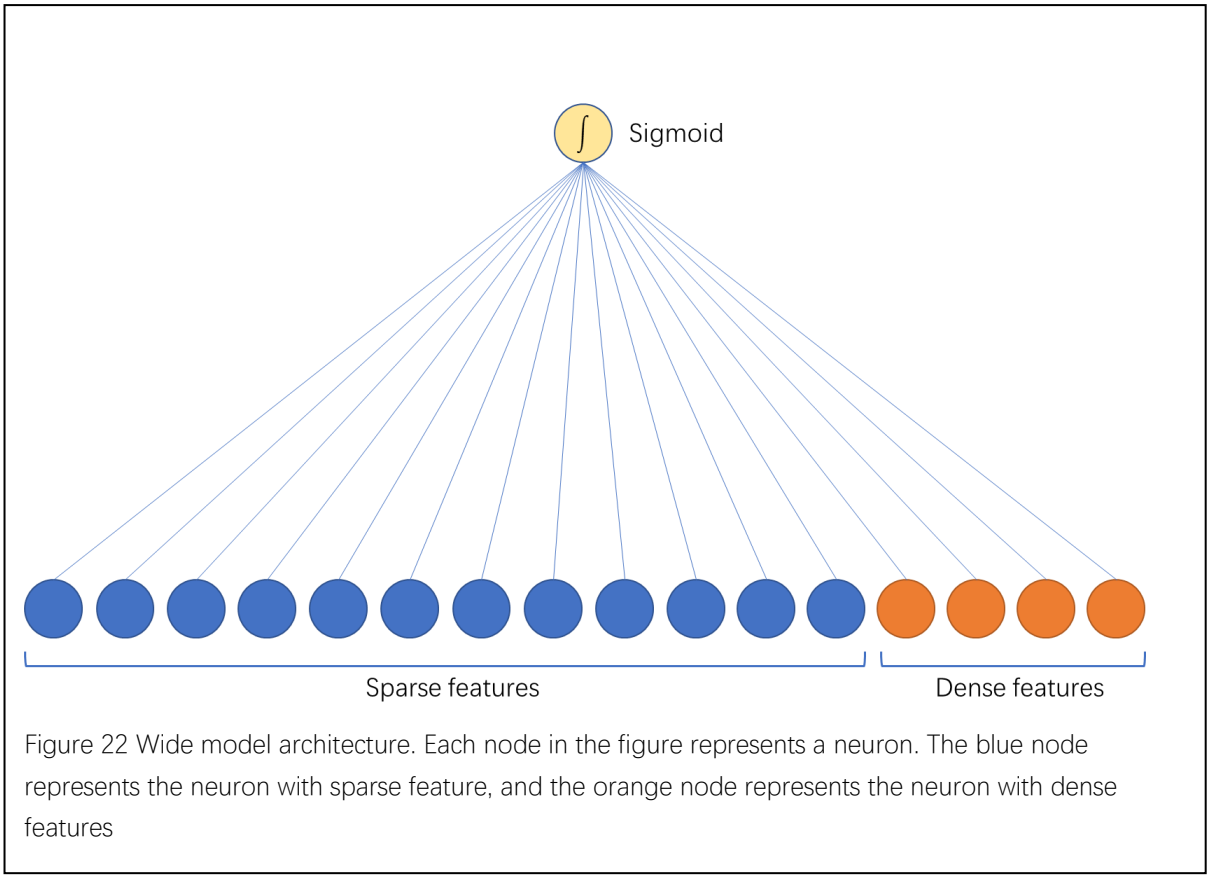

Figure 22 Wide model architecture. Each node in the figure represents a neuron. The blue node represents the neuron with sparse feature, and the orange node represents the neuron with dense features

5. The outputs of the embedding layer go through a sigmoid function

6. Set the regularization coefficient for the model.

7. Set loss function at the end of the architecture with regularization.

Figure 22 shows the overall architecture of the wide model and Figure 23 shows the flow diagram of the wide model in Tensorflow.

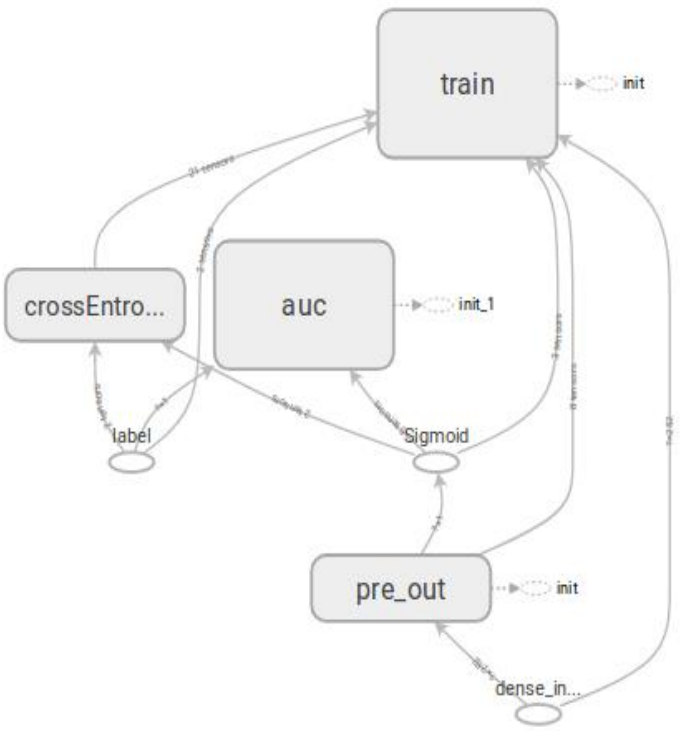

Figure 23 Flow diagram of the wide model. Each box in figure represents a data processing center. After all neuron data (crossEntropy, auc, pre_out) is processed, they are sent to the training processor to train the model.

5.4 Wide model training

The training process was also implemented on Tensorflow framework(67):

1. Randomly sort the samples to make sure patients who diagnosed with cancer not close to each other.

2. Divide the training dataset with several batches.

3. Fit the copy number variation data and diagnosis into the “placeholder.”

4. Start training by optimizing the loss function with different regularization coefficient using gradient descent.

5. Report evaluation matrix every iteration.

5.5 Results of wide model

In this study, we choose Area Under the Curve (AUC) to be the evaluation criteria. Figure 24 shows the learning curve of the best performance AUC on the test set. In the wide model, we have 25 dense inputs and 250 sparse inputs and trained with 10000 iterations and a 0.01 learning rate.

The learning curve shows that the model was not overfitting so much since the AUC value when the learning curve converges is similar to the value on the test dataset. Figure 25 and Table 8 showed the AUC value on the test dataset when we applied different L2 regularization coefficient. There is no significant change between differentnt L2 regularization coefficient, which also proves that the wide model was not overfitting. These results are reasonable because the wide model is a relatively small-scale model and it is hard to overfit.

Table 8 AUC performance on test dataset with different regularization coefficient

Regularization coefficient	1	2	3	4	5
l2=0	0.779782	0.754973	0.721669	0.747027	0.758383
l2 = 0.01	0.729618	0.768639	0.773171	0.784229	0.719601
l2 = 0.03	0.752169	0.711095	0.729579	0.726009	0.771395
l2 = 0.05	0.782318	0.731056	0.742412	0734613	0.723894

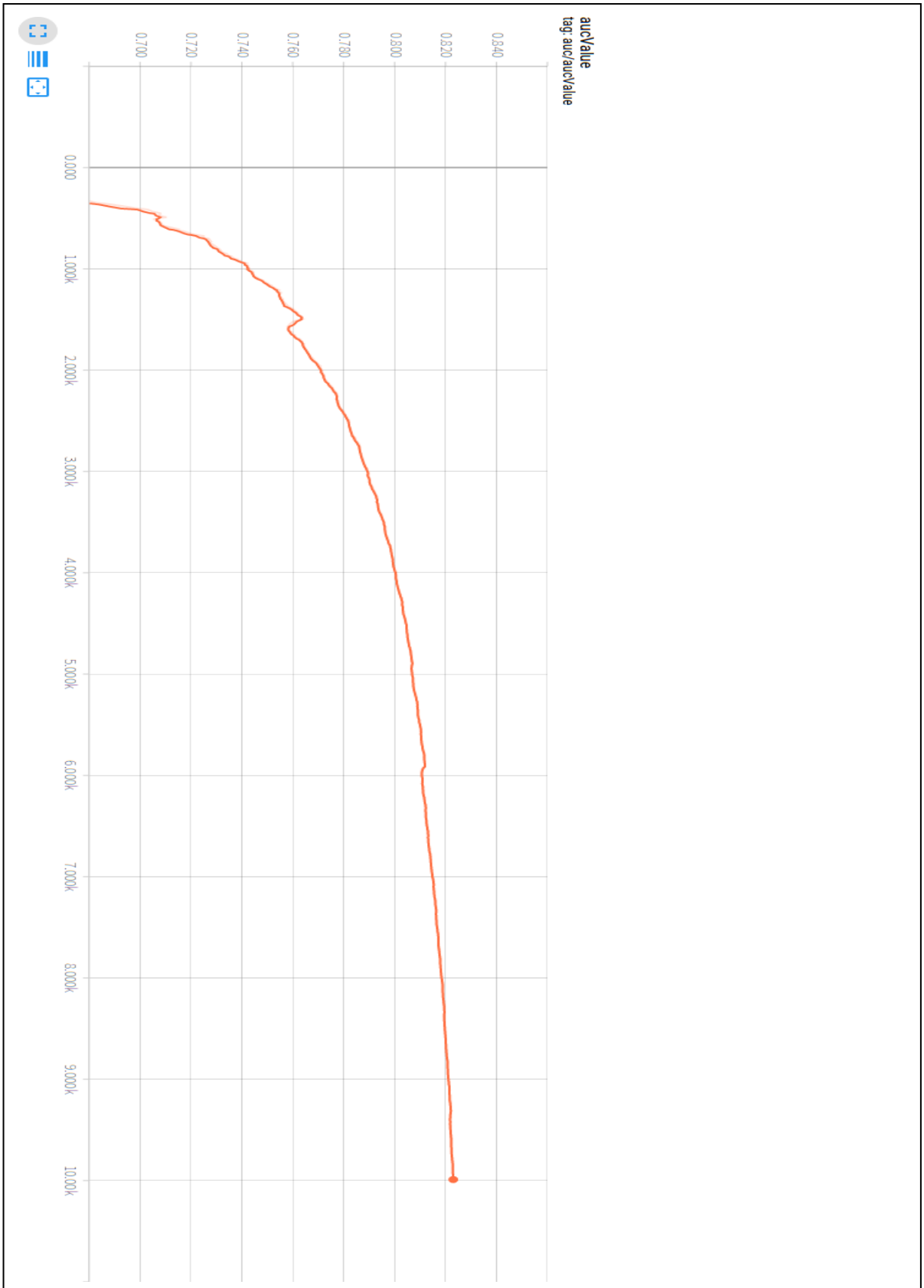

Figure 24 The learning curve of the wide model. The X-axis represents the training iterations (the training time) and the Y-axis represents the AUC value. We can see the model is learning with the AUC value going up as training time goes on.

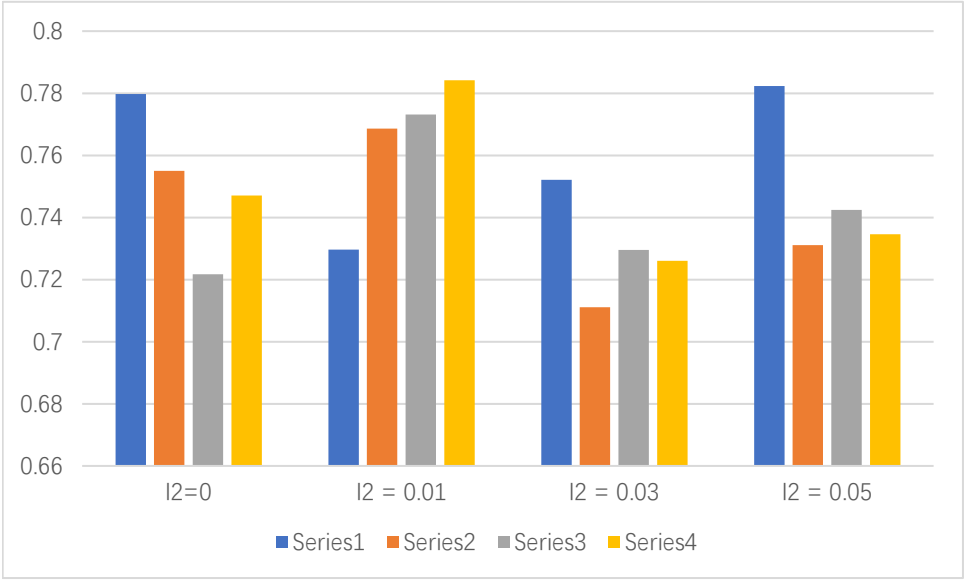

Figure 25 AUC performance on test dataset with different regularization coefficient. Different color on the bar represents the different model training with the same setting. There is no significant difference between different regularization coefficient.

To better understand the learning processing during training, the parameters in the neural network was plotted using Tensorboard(67). Figure 26 shows the change of the weight distributions over time for each neuron in a histogram visualization. The x-axis is the value of the weights, Y-axis is the number of training iterations (training time), and the Z-axis shows how the weight value densely clustered in the weight value space. For example, each X-Z place shows a weight histogram for each iteration. The diagram is like an audio waveform spread out over time and reflects the trend of the parameter change during training(67). Figure 27 shows the change in weight distribution over time in a different view. The palest line shows the maximum value of the weights in each iteration. With the color goes darker, the line represents the value that covers 90th percentile weights, 60th percentile weights, and so on. A diagram in this view shows how densely clustered for a parameter during training over time. Similarly, Figure 28 shows the change

of bias over time. The distribution could not be plotted since the wide model is a single layer model, and it just has one bias for each iteration.

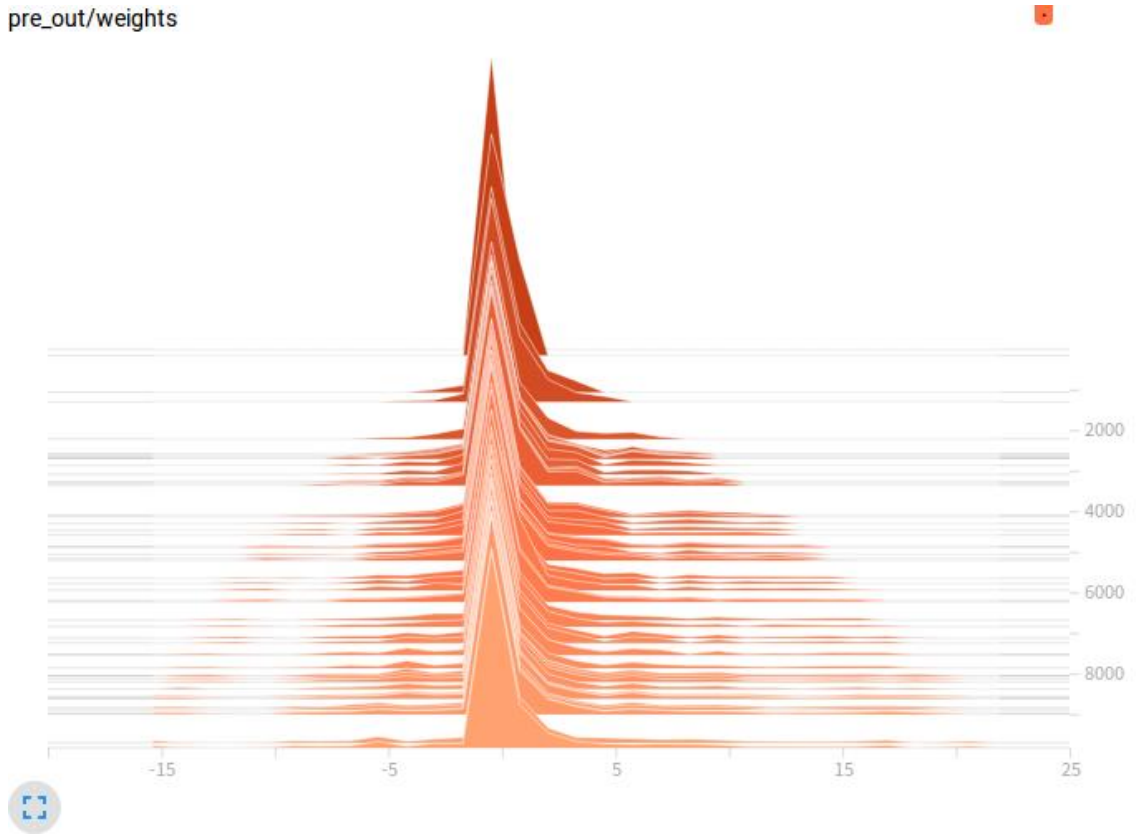

Figure 26 Wave plot of the weight distribution changes over time. X-axis represents the parameter value, and the Y-axis represents the training iteration (training time). Each cross section represents the parameter distribution in each training iteration. We can monitor the value shift over time.

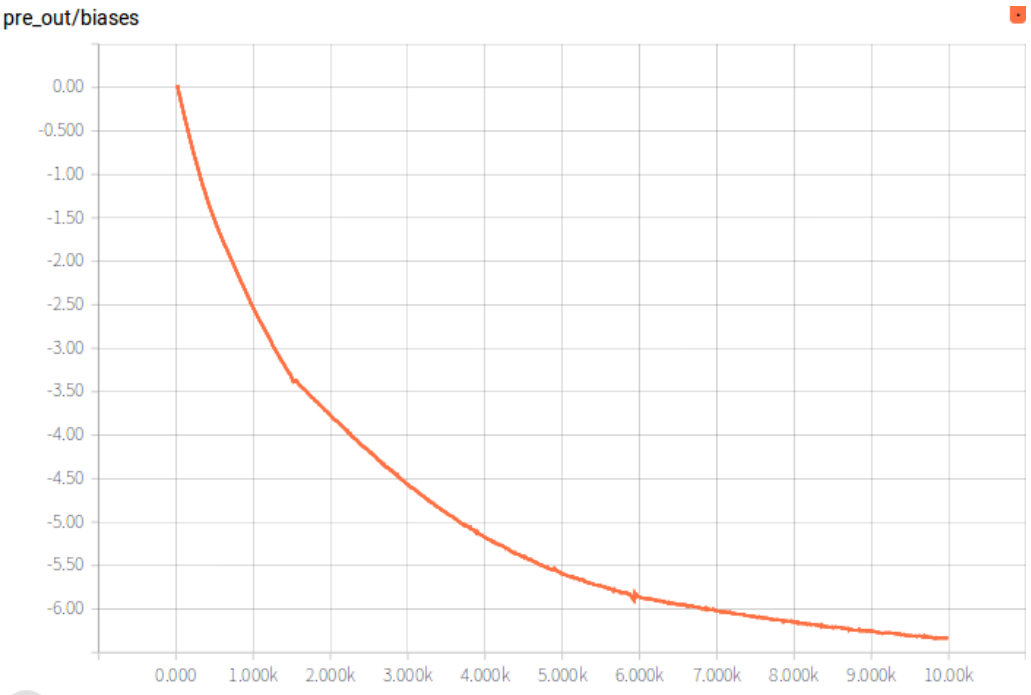

Figure 28 Bias change over time. X-axis represents the bias value, and the Y-axis represents the training iteration (training time). With the training goes on, the bias decrease over time.

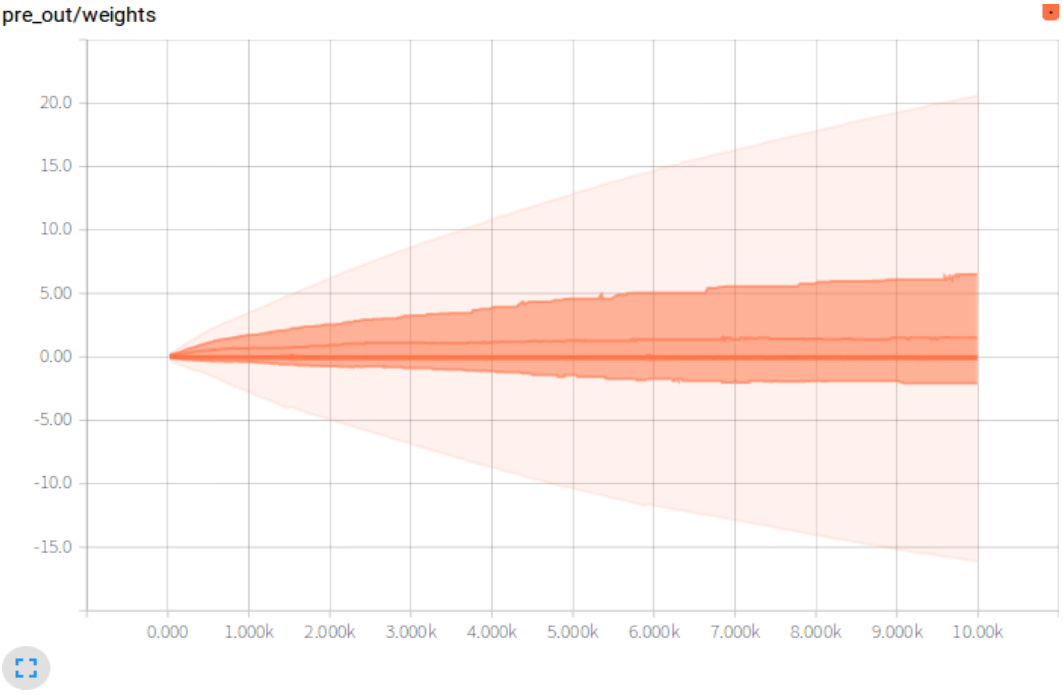

Figure 27 Cluster plot of the weight distribution changes over time. Y-axis represents the parameter value, and the X-axis represents the training iteration (training time). Darker color represents more parameters falling within the darker area. The parameters is spreading out during training.

The weights spread out over time in the diagram. The weights were set to be very close

to zero initially. So, they were clustered in the beginning. Since the model is learning relatively well, the distribution spread out gradually and smoothly. Moreover, the bias decrease over time from the diagram, which also supports that the model learns the pattern smoothly.

Chapter6 Deep model for colorectal cancer prediction

6.1 Deep model

The word “deep” in the deep model refers to the multiple layers of neural network architecture. The input data is transformed through all the layers. As mentioned before, the deep model is widely used in many machine learning problems. The deep model can perform well with complex non-linear relationships. A deep model could be expressed as a complex combination of many single layer “wide model.” The extra layers can make the model compute the hidden features from lower layers, model complex data with less computing time and space than shallow architectures. Thus, the deep model can be able to process more complex data the wide model(68).

However, the deep model tends to be overfitting because the extra layers allow the model to do some “extra jobs” for the rare dependencies in the training data. In the training process, a deep model requires the adjustment of many training parameters, such as the number of layers, the number of neurons in the hidden layer, and the coefficient of regularization. In this study, we compared the results on the test dataset with different training hyper-parameters.

The feature engineering in deep modeling follows the same process as the wide model in this study.

6.2 Deep model architecture

The whole architecture was also built based on Tensorflow framework(67). The architecture is built based on the following steps:

1. Build a function of embedding layers with weights and bias.

2. Build a function of fully connected layers with weights, bias, and activation function.

3. Set the number of layers of the model.

4. Set the number of neurons for the first layer, and distribute “placeholder” for the neurons

5. Set the number of neurons for the hidden layers.

6. Connect the input with the fully connected layer with activation functions

7. Connect the neurons in hidden layers with fully connected layers with activation functions.

8. Set the regularization coefficient for the model.

9. Set loss function at the end of the architecture with regularization.

10. Figure 29 shows the overall architecture of the deep model and Figure 30 shows the flow diagram of a four layer deep model in Tensorflow.

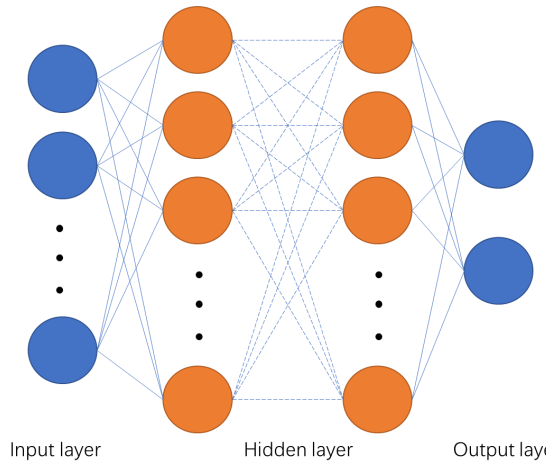

Figure 30 Deep model architecture. Each node in the figure represents a neuron. The blue layer represents the input layer and output layer. The orange layer represents the hidden layer. The connections between layers is shown with dotted line. The connections between hidden layers could go through multiple layers.

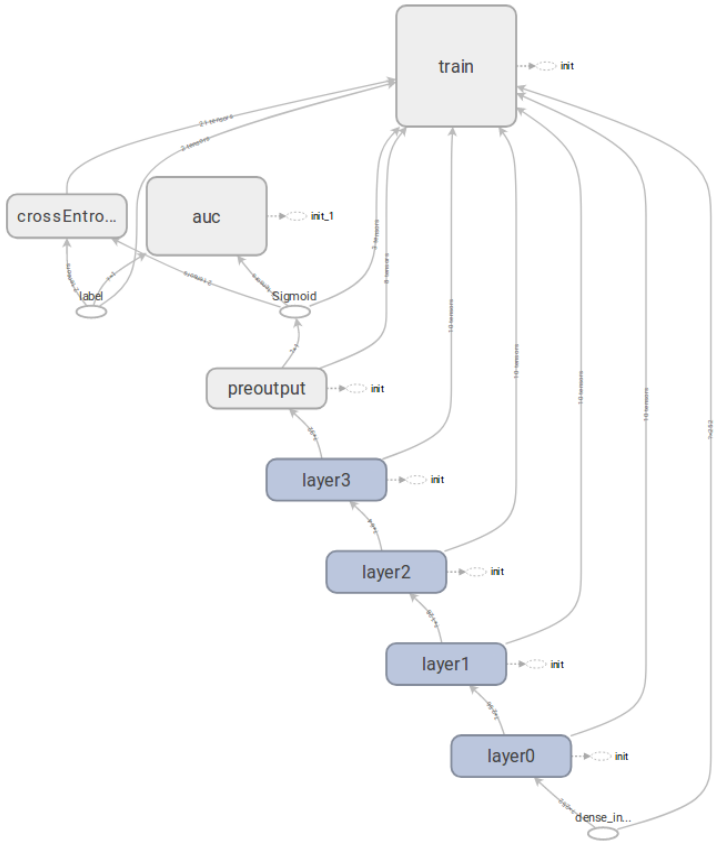

Figure 29 Flow diagram of the wide model. Each box in figure represents a data processing center. After all neuron data (layers, crossEntropy, auc, preoutput) is processed, they are sent to the training processor to train the model.

6.3 Deep model training

The training process was also implemented on Tensorflow framework(67):

1. Randomly sort the samples to make sure patients who diagnosed with cancer not close to each other.

2. Divide the training dataset with several batches.

3. Build the overall architecture with hidden layers.

4. Feed the copy number variation data and diagnosis into the “placeholder.”

5. Computed the value for neurons in each layer.

6. Start training by optimizing the loss function with different regularization coefficient using gradient descent.

7. Report evaluation matrix every iteration.

8. Repeat the process above with different network architectures.

6.4 Results of deep model

The Area Under the Curve (AUC) was also used to be the evaluation criteria for the deep model. Figure 33 shows the learning curve of the best performance AUC on the test set. In the deep model, we have 25 dense inputs and 250 sparse inputs and trained with 500 iterations and a 0.01 learning rate. The deep model was trained with fewer iterations than the wide model because the model is prone to overfitting. A smaller number of iterations could decrease the probability with which the model learns the rare dependencies.

The learning curve shows that the model was overfitting since the AUC value when the learning curve converges is much larger than the value on the test dataset. Table 9 and Figure 32 showed AUC value on the test dataset when we applied different network architecture. The four hidden layer architecture with a gradually decreasing number of neurons perform best in this model. Table 10 and Figure 31 showed AUC value on the test dataset when we applied the regularization coefficient. The regularization did not work very well, probably because the available patient sample data was too small to avoid the deep model to learn rare dependencies.

Table 9 AUC performance on test dataset with different network architecture

Network architecture	1	2	3	4	5
250-256-128-64-32	0.663593	0.690123	0.628179	0.682243	0.671184
250-256-128-32-32	0.604324	0.583023	0.618644	0.560698	0.632144
250-256-128-32	0.584442	0.603061	0.607788	0.508004	0.610845

Table 10 AUC performance on test dataset with different regularization coefficient

Regularization coefficient	1	2	3	4	5
l2 = 0	0.663539	0.690122	0.628185	0.682234	0.671178
l2 = 0.01	0.619053	0.685714	0.68335	0.641879	0.698381
l2 = 0.2	0.623711	0.694759	0.659038	0.680681	0.640031

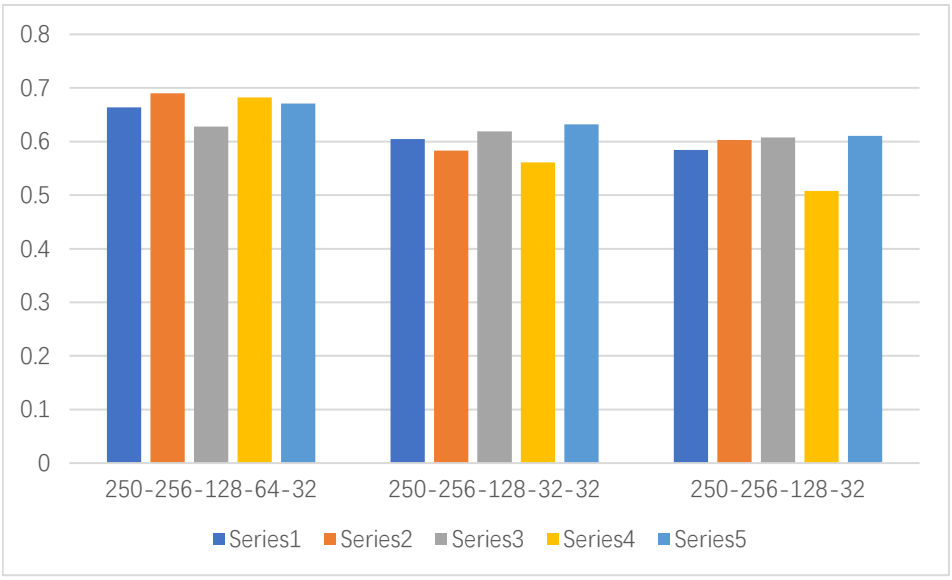

Figure 32 AUC performance on test dataset with different network architecture. The number in each architecture "xx-xx-xx-xx" represents the number of nodes in each layer.

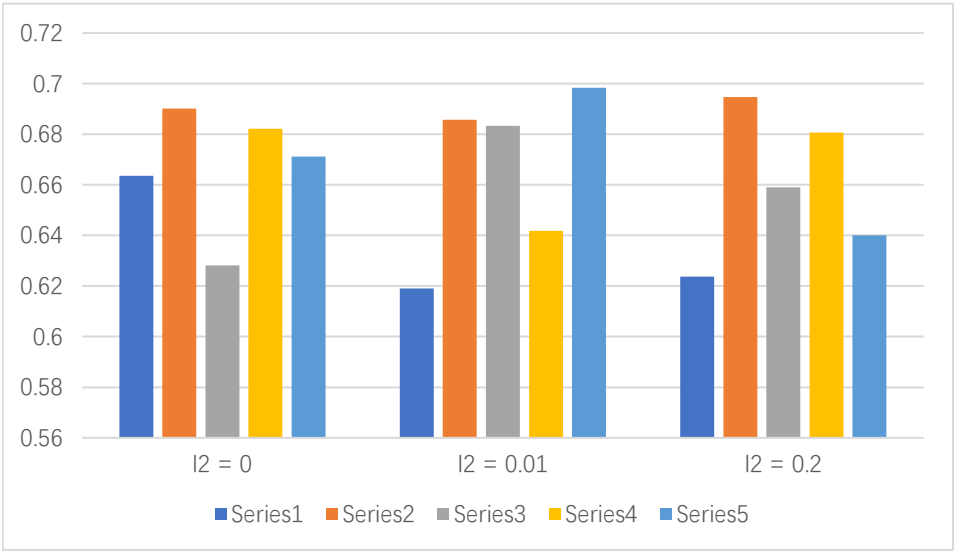

Figure 31 AUC performance on test dataset with different regularization coefficient. There is no significant difference between different regularization coefficient. Different color on the bar represents the different model training with the same setting.

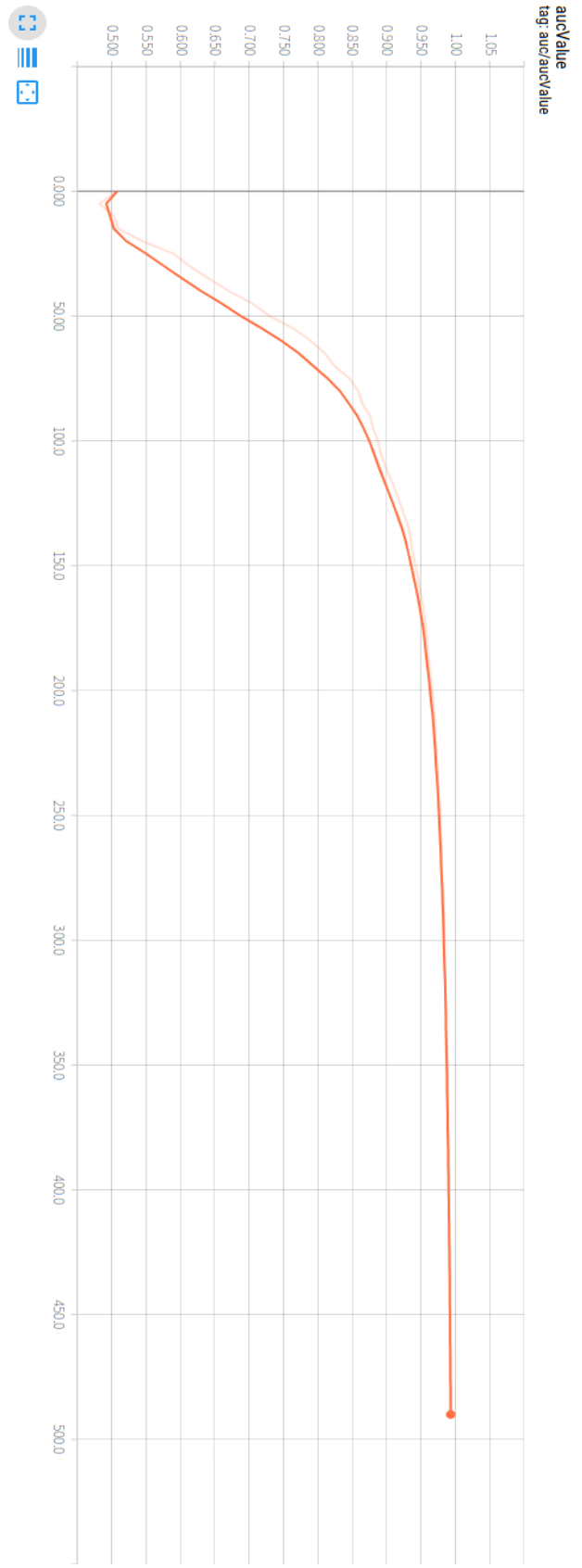

Figure 33 The learning curve of the deep model. The X-axis represents the training iterations (the training time) and the Y-axis represents the AUC value. We can see the model is learning with the AUC value going up as training time goes on.

Figure 34 Cluster plot of the weight distribution, bias distribution, and activation distribution changes over time. Y-axis represents the parameter value, and the X-axis represents the training iteration (training time). Darker color represents more parameters falling within the darker area. The parameters is spreading out during training.

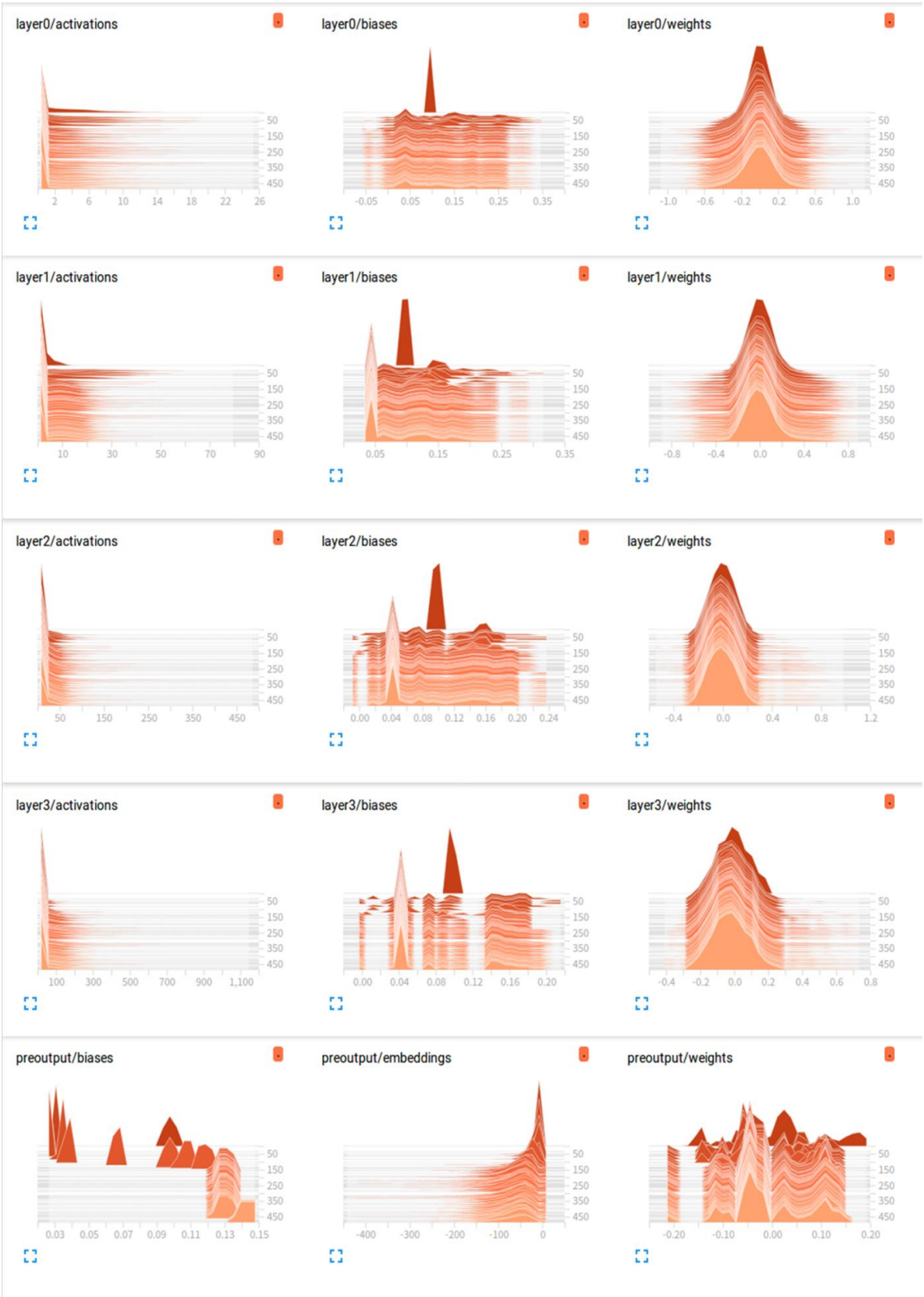

Figure 35 Cluster plot of the weight distribution, bias distribution, and activation distribution changes over time in deep model. X-axis represents the parameter value, and the Y-axis represents the training iteration (training time). Each cross section represents the parameter distribution in each training iteration. We can monitor the value shift over time.

Figure 34 shows the cluster plot of the weight distribution, bias distribution, and activation distribution changes over time. Figure 35 shows the wave plot of the weight distribution, bias distribution, and activation distribution changes over time. The first four rows in these two figures showed the distributions of the parameters in the four hidden layers. The last row in these two figures showed the distribution of the parameter in the last layer, which is similar to the diagram of the wide model.

The relatively smooth wave plot and the smooth learning curve prove the model was not broken. However, the concave at the 100th iteration reflects the abnormal learning occurred. This concave could explain the deep model was learning the rare dependencies after the 100th iteration, which probably cause the overfitting of the deep model. However, the training with only 100 iterations was also performed, but the AUC performance does not have much improvement.

Chapter7 Dense-sparse model for colorectal cancer prediction

7.1 Dense-sparse model

In the cancer prediction study, the copy number variation data for many segments from a patient can be unavailable. This situation is very usual in the TCGA dataset, which causes the data used for training very sparse. As mentioned before, very sparse training data can cause many problems for neural network training. Also, the deep model and wide model do not take the low-order or high-order feature interactions into consideration. However, in a real situation, gene segments could interact with each other and affect the traits together. By introducing the feature interactions into neural network architecture, the model could "predict" the unavailable features using the interactions it learned during training. In this section, we show an end-to-end learning model that could incorporate feature interactions to deal with the sparse copy number variation data.

The critical problem here is to model the feature interactions effectively since many interactions are hidden in the complex dataset, which could only be identified by machine learning automatically. Even some of the genetic interactions between gene segments could be determined by experts, but it could be a massive workload, especially when the number of features is large. Factorization Machines(69) was designed for feature interactions by an inner product of latent vectors. DeepFM(70) combined deep neural networks with factorization machine and showed promising results on the large sparse dataset. In this study, a dense-sparse model is reported based on the inner product theory to deal with the feature interactions.

The dense-sparse model processes the dense features and sparse features separately first and combines them later to go through a deep neural network. Figure 36 shows the overall architecture of the model. The dense features go through a deep neural network and output a fixed size vector. The sparse features divide into several segments first. Each segment goes through an embedding layer and output a vector with the same size of the vector that output from the dense feature. Then, pairwise dot products by the previous outputs were concated to a large vector (Figure 37). During this process, the interactions between features are incorporated into the model. This large vector then goes through another deep neural network to predict the final results. Figure 38 shows

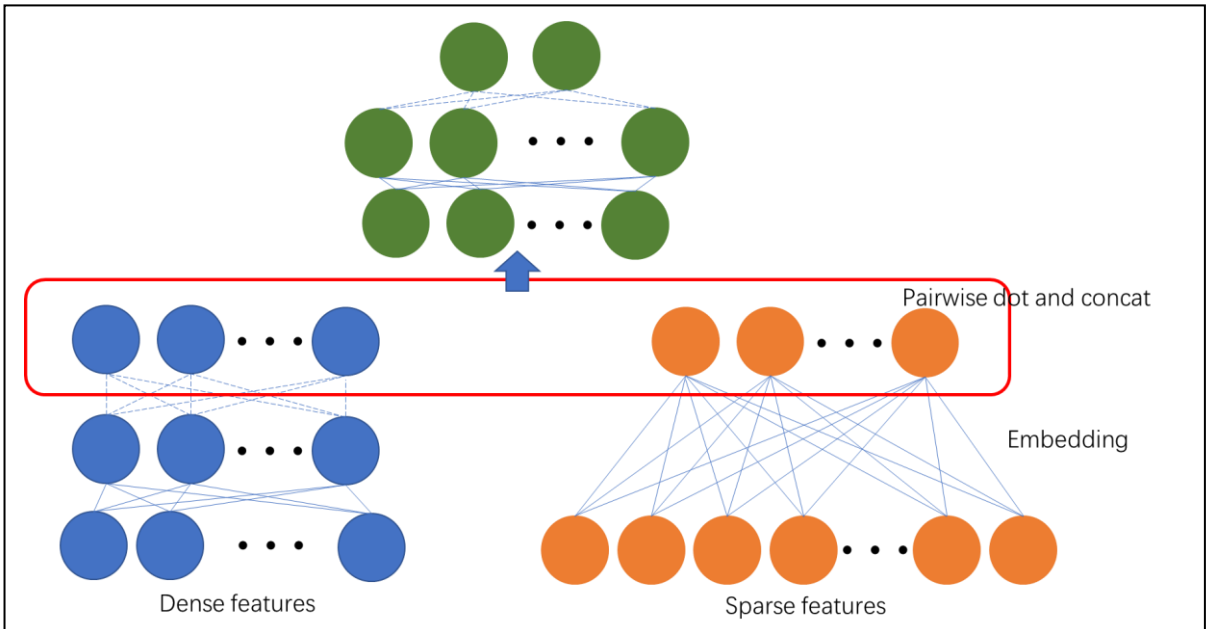

Figure 36 Dense-sparse model architecture. Each node in the figure represents a neuron. The blue nodes together represent network which process the dense features. The orange nodes together represent the network which process the sparse features. The green nodes together represent the network which process the data after pairwise dot product and concat.

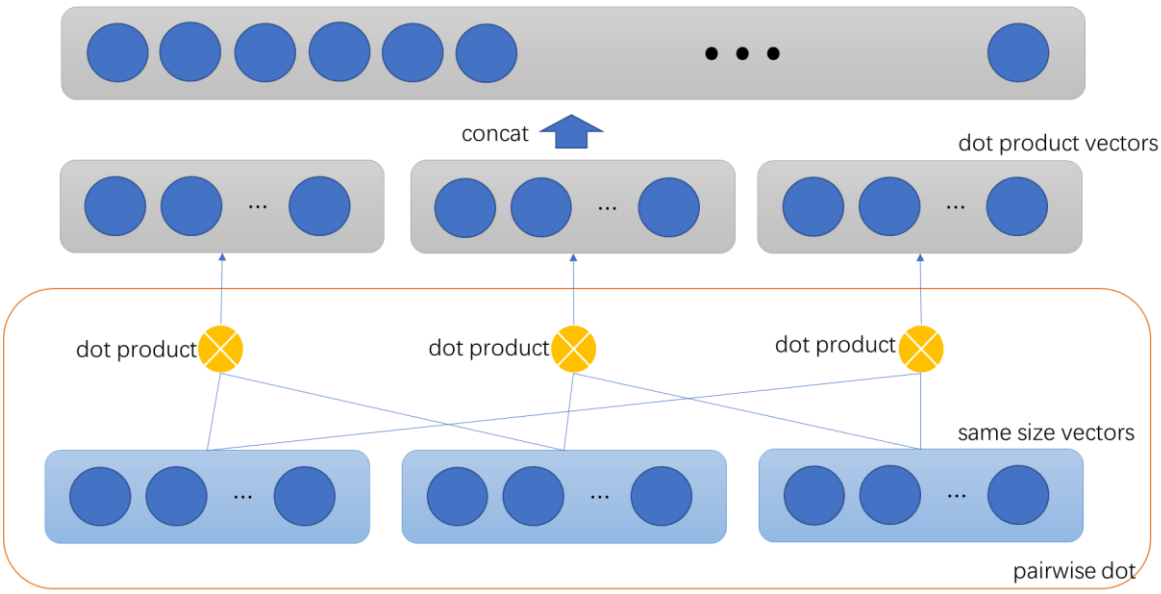

Figure 37 Vector pairwise dot and concat process. The dot product operation represents the dot product of the same size vector. The concat operation represents the connection of each vector in the first axis.

The flow chart after the vectors concated together (the whole flow chart is sophisticated so only the latter part of the graph is shown here). The figure shows the vector pairwise dot and concat process.

The whole architecture was also built based on Tensorflow framework(67). The architecture is built based on the following steps:

1. Build a function of embedding layers with weights and bias.

2. Build a function of fully connected layers with weights, bias, and activation function.

3. Set the number of layers for the dense features, sparse features, and concat vector.

4. Set the number of neurons for each layer, and distribute “placeholder” for the neurons

5. Connect the dense input with the fully connected layer with activation functions

6. Connect the sparse input with the fully embedding layer.

7. Connect the neurons in hidden layers for all the architecture.

8. Set the regularization coefficient for the model.

9. Set loss function at the end of the architecture with regularization.

7.2 Dense-sparse model training

The training process was also implemented on Tensorflow framework(67):

1. Randomly sort the samples to make sure patients who diagnosed with cancer not close to each other.

2. Divide the training dataset with several batches.

3. Build the overall architecture with hidden layers.

4. Feed the sparse features to the input “placeholder” for the sparse network.

5. Feed the dense features to the input “placeholder” for the dense network

6. Computed the value for neurons in each layer.

7. Start training by optimizing the loss function with different regularization coefficient using gradient descent.

8. Report evaluation matrix every iteration.

9. Repeat the process above with different network architectures.

7.3 Results of dense-sparse model

The Area Under the Curve (AUC) was also used to be the evaluation criteria for the dense-sparse model. Figure 39 shows the learning curve of the best performance AUC on the test set. In the dense-sparse model, the 25 dense inputs go through a neural network with one hidden layer (16 neurons) and output a vector of fixed length 16. The 250 sparse inputs are divided into five fields. Each field goes to an embedding layer and outputs a vector of fixed length 16. After pairwise dot and concat, the long vector goes through a neural network with two hidden layers, which has 128 neurons and 16 neurons respectively. The model was trained with 1000 iterations and a 0.01 learning rate. A 0.703624 AUC was reported and cannot be improved by changing the architecture or the regularization coefficient.

The learning curve also shows that the model was overfitting since the AUC value when the learning curve converges is much larger than the value on the test dataset. Figure 40 shows the cluster plot of the weight distribution of each layer, bias distribution of each layer, dense activation, and dot product distribution changes over time. Figure 41 shows the wave plot of the weight distribution of each layer, bias distribution of each layer, dense activation, and dot product distribution changes over time.

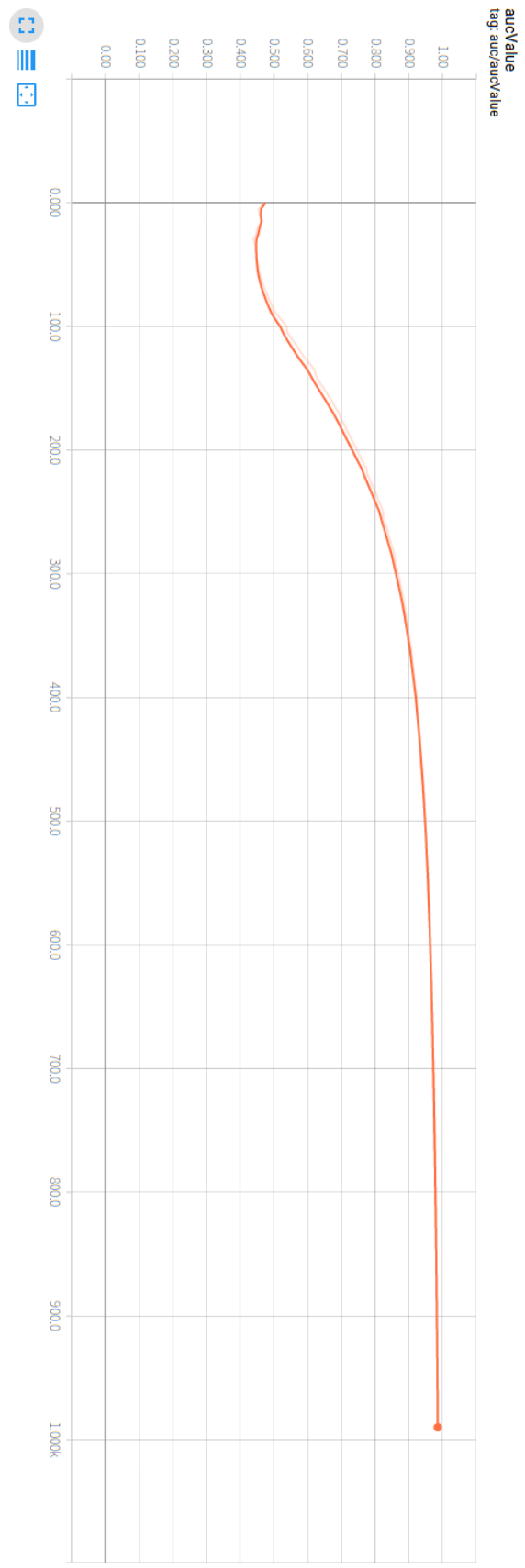

Figure 38 The learning curve of the dense-sparse model. The X-axis represents the training iterations (the training time) and the Y-axis represents the AUC value. We can see the model is learning with the AUC value going up as training time goes on.

Figure 39 Cluster plot of the weight distribution, bias distribution, and activation distribution changes in dense-sparse model over time. Y-axis represents the parameter value, and the X-axis represents the training iteration (training time). Darker color represents more parameters falling within the darker area. The parameters are spreading out during training.

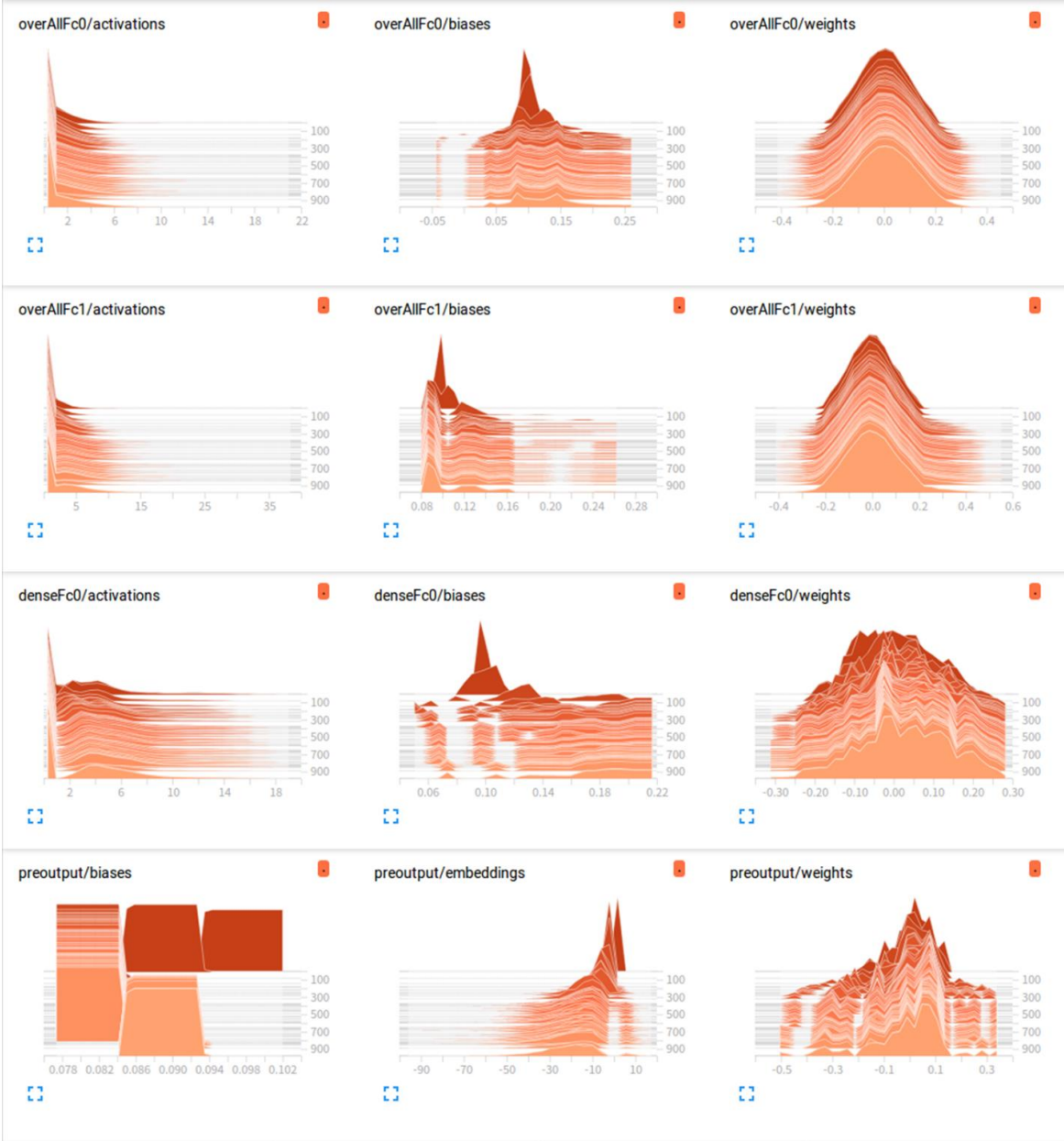

Figure 40 Cluster plot of the weight distribution, bias distribution, and activation distribution changes over time in dense-sparse model. X-axis represents the parameter value, and the Y-axis represents the training iteration (training time). Each cross section represents the parameter distribution in each training iteration. We can monitor the value shift over time.

The relatively smooth wave plot and the smooth learning curve prove the model was not broken. However, the abnormal curve of the first overall layer and the pre-output layer shows that the model is probably overfitting. From the wave plot, the distribution of the

parameters tends to spread out, which may reflect the model are learning rare independencies. Especially, the wave plot of the bias in pre-output shows a significant abnormal shape, which potentially indicates that the model is significantly overfitting at the end of the training.

.

Chapter8 Pairwise model for colorectal cancer prediction

8.1 Pairwise model in learning to rank

Learning to rank is a concept from a machine learning system. Traditional machine learning problems predict a single instance at a time. For example, we feed all the features associated with copy number variation from a single patient in the previous section. The object of traditional machine learning is to estimate a numerical score for that single instance. However, learning to rank can solve problems in several instances. The object of learning to rank is to predict an order for these instances. Especially, when applying to two instances, it calculates the "distance" between the two instances.

This concept of calculating the similarity can usually be found in image recognition. Traditionally, researchers model the image similarity by category level(71)(72). Two images are considered to be similar if they are from the same category. However, this method has limitations on distinguishing images in the same category. Another way to model the image similarity is to extract hidden features like SIFT(73) and HOG(74). The success of these methods highly depends on the features extracted. Recently, A deep learning model for learning the Fine-grained image similarity was reported(75). The deep learning model could be able to learn the hidden features jointly and has a good performance on image similarity ranking. Instead of calculating the similarity between images, we calculate the similarities of genetic data (copy number variation) between two patients using the triplet loss(75) in this study.

The model takes two patients' data as input and calculates the similarity between the two patients. In this study, we set the one patient as a positive sample (patient who is diagnosed with cancer) and the other patient as a negative sample (patient who is not diagnosed with cancer) as shown in Figure 42 a). A typical feature vector is shown in Figure 42 b). If the first patient is a positive sample, the label l of this feature vector is set to be one. If the second patient is a positive sample, the label l of this feature vector is set to be zero.

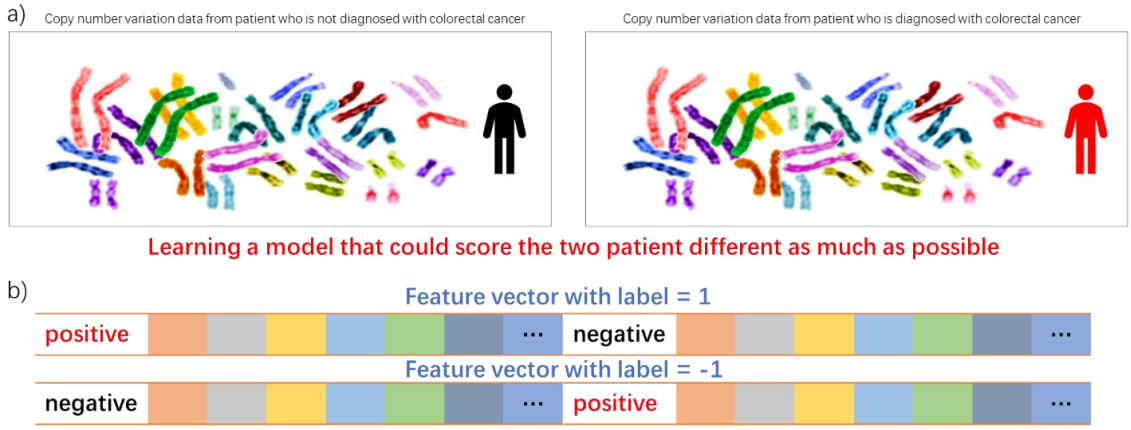

Figure 41 Sample data pairs used for training. Figure 42 a) shows two patients' genome, one diagnosed as colorectal cancer, the other did not. Figure 42 b) shows two typical feature vectors fed into the pairwise model.

We define the model score the first patient as S_1, the second patient as S_2. Then we define the rank loss function for training as:

$$l(p_1, p_2) = max\{0, m + l * (S_2 - S_1)\}$$

Where p_1 and p_2 are the genetic data from the patient pair, m is the margin that the ideal smallest difference of the score could be, l is the label of the feature vector. Using

the rank loss here, a model to calculate the similarity of the genetic data is built as Figure 43. Figure 44 shows the data flow chart in Tensorflow(67).

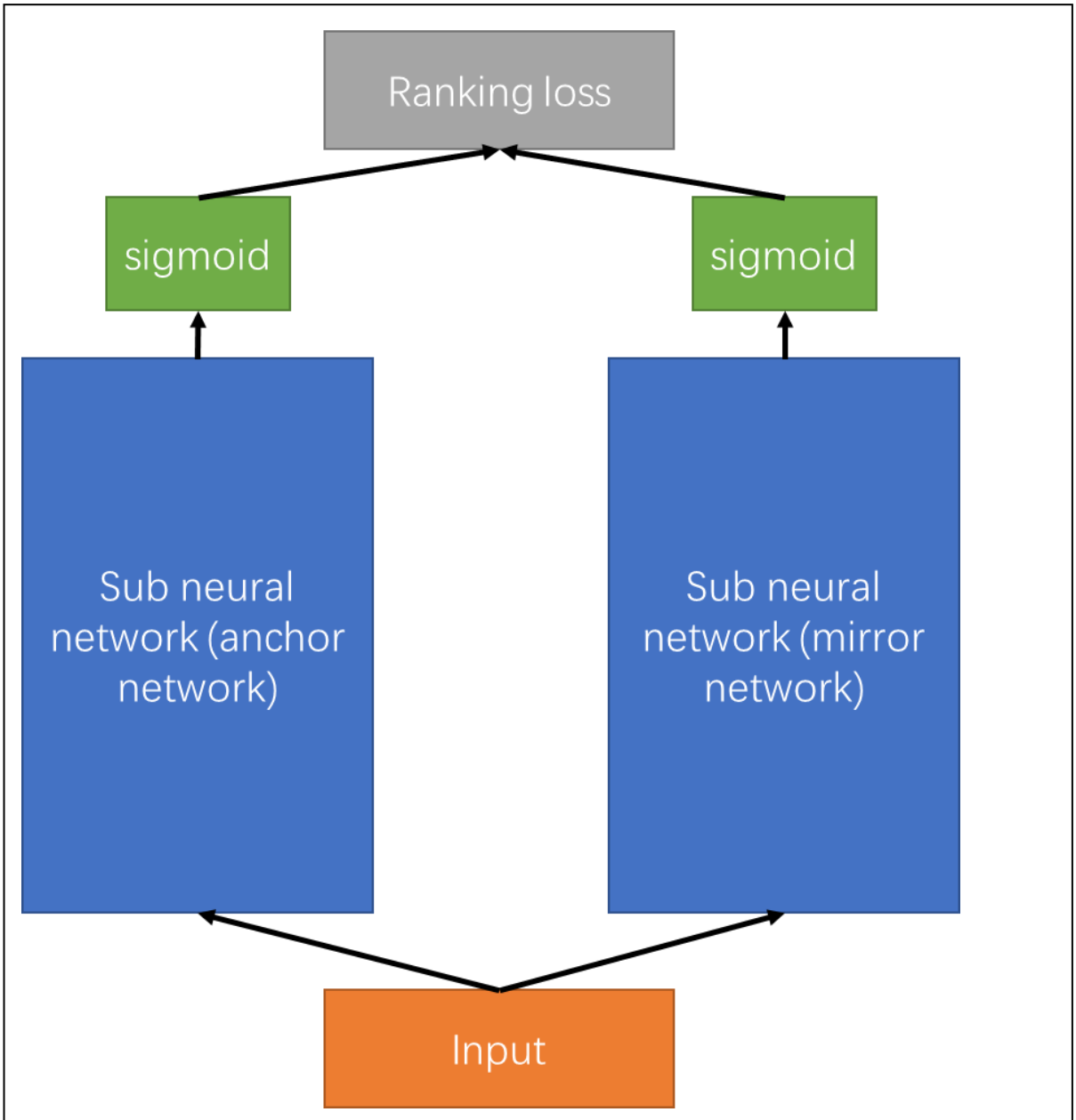

Figure 42 Basic architecture for a pairwise model. One patient's data from the input feature vector go through the anchor network, another patient's data from the input feature vector go through the mirror network. After a sigmoid function, the scores S_1 and S_2

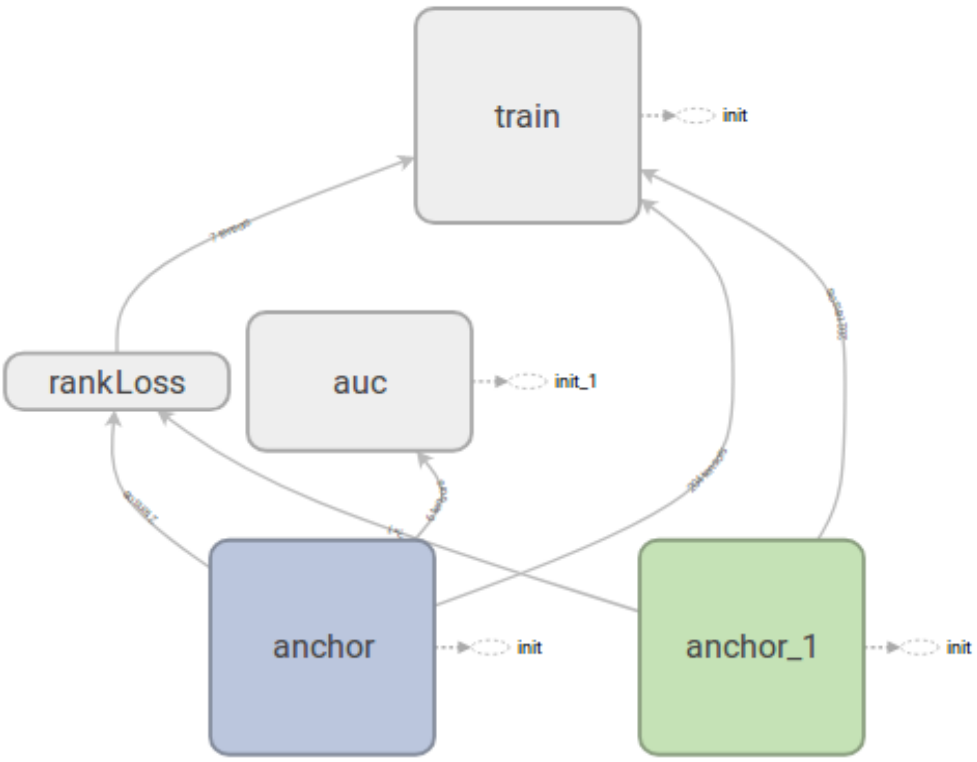

Figure 43 Flow diagram for pairwise model (anchor_1 refer to mirror in Tensorflow). Anchor and anchor_1 represent two identical neural networks with different input data. Anchor is fed with positive data and anchor_1 (mirror) is fed with negative sample. All the data was calculated and goes to training processor finally.

The sub neural network in Figure could be any forms. From the results in the previous section, the wide model performs best in our study since it is not too complicated and not overfitting. So, the wide model is used as the sub neural network in the pairwise model.

As the feature processed in Gradient Boost Machine, a feature vector for each patient was extracted by each patient. In the pairwise model, we do an extra step to pair the

patients as shown in Figure 42. A random shuffling process was also performed after the paring.

The whole architecture was also built based on Tensorflow framework(67). The architecture is built based on the following steps:

1. Build a function of embedding layers with weights and bias.

2. Build a function of fully connected layers with weights, bias, and activation function.

3. Set the number of neurons for the sub wide neural networks and distribute "placeholder" for the neurons

4. Connect the neurons with a sigmoid function.

5. Connect the two sub-neural networks with rank loss function.

8.2 Pairwise model training

The training process was also implemented on Tensorflow framework(67):

1. Randomly shuffle the new vectors to make sure the label of the vector not repeat.

2. Divide the training dataset with several batches.

3. Build the overall architecture with the rank loss

4. Feed the first features to the anchor sub neural network.

5. Feed the second features to the mirror sub neural network.

6. Computed the value for neurons in each layer.

7. Start training by minimizing the loss function using gradient descent.

8. Report evaluation loss every iteration.

9. Repeat the process above with different network architectures.

8.3 Results for pairwise model

The Area Under the Curve (AUC) was also used to be the evaluation criteria for the pairwise model. Figure 45 shows the learning curve of the best performance AUC on the test set. In the pairwise model, the 275 sparse and dense features from one patient go through a wide model with a single layer, another 275 features from another patient go through a mirror wide model with the same parameter as the previous one. The model was trained with 8000 iterations and a 0.01 learning rate. Because there are more than 2000000 feature vectors due to paring. The margin is set to be 0.1 to get the best performance. The model with the best performance reports a 0.802655 AUC value.

The learning curve also shows that the model was overfitting since the rank loss is very small when the learning curve converges. Figure 46 shows Wave and cluster plot of the weight distributions of the anchor network, bias distribution of the anchor, embedding (outputs before sigmoid function) distribution of the anchor network. The wave plot and scatter plot are the smoothest one compared to the previous model, which means the model is learning patterns smoothly and not overfitting easily. The aim of the pairwise model is to push the predictions of a positive sample and a negative sample away. Therefore, the AUC performance of the pairwise model is the best compared to the previous model. Figure 45 shows a comparison between the models in this study.

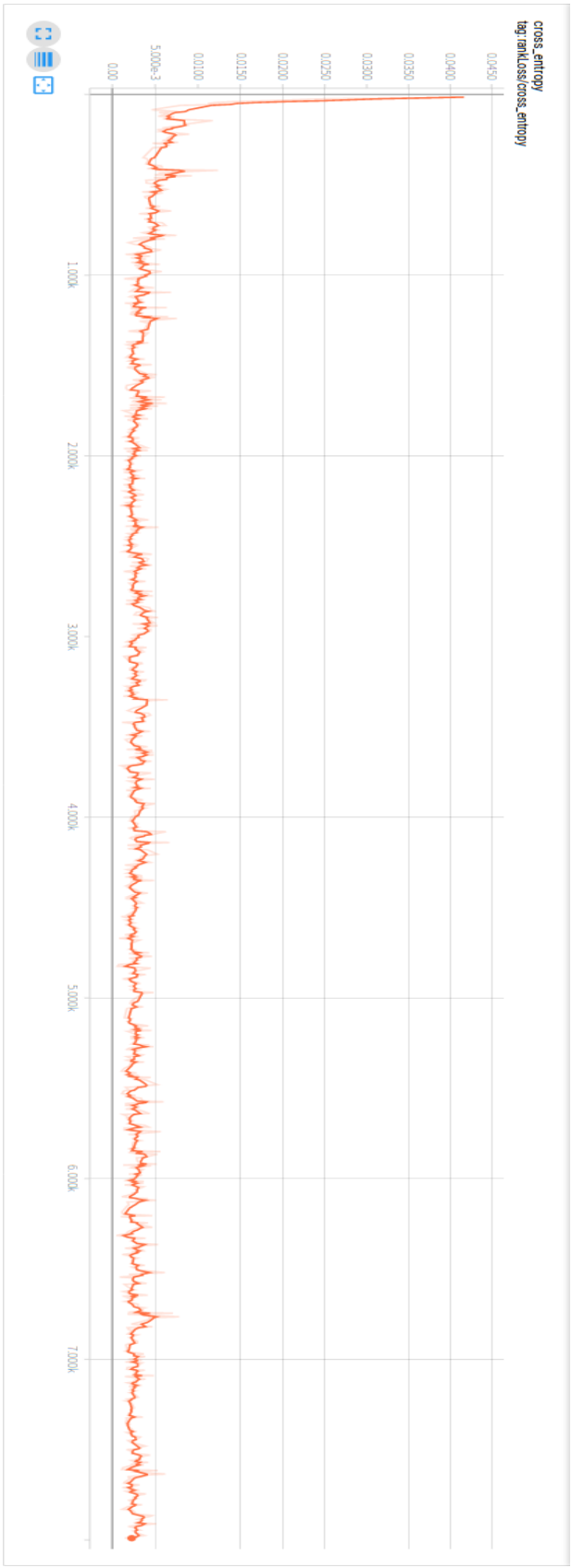

Figure 44 Learning curve of the pairwise model. The X-axis represents the training iterations (the training time) and the Y-axis represents the AUC value. We can see the model is learning with the AUC value going up as training time goes on.

Figure 46 Wave and cluster plot of the weight distributions, bias distribution of the ancho, embedding (outputs before sigmoid function) distribution of the anchor network. The distributions of the mirror neural network are the same since the two sub neural networks share the same parameter.

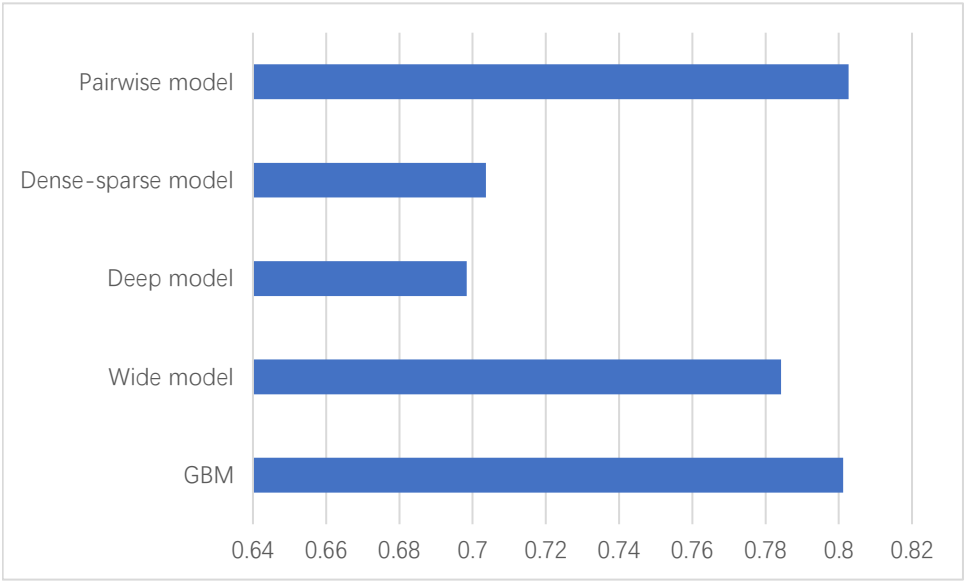

Figure 45 AUC performances for all the models in this study

Chapter9 Summary

In this thesis, we have discussed the principle of artificial intelligence in medicine. Starting from the predictions in medicine, evaluation criteria, sensitivity, and specificity, the basic machine learning process was derived. With describing the two main classes of machine learning, supervised machine learning and unsupervised machine learning, predicting the probability of getting colorectal cancer was introduced, and many recent machine learning techniques were performed.

We have conducted many machine learning methods to estimate the probability of getting colorectal cancer. First, the Gradient Boosting Machine (GBM) was used to predict the probability. The mathematical algorithm of GBM was derived and performed. Also, an Area Under the Curve (AUC) value was reported to evaluate the performance of the model. The distribution of the predictions (Figure 11) proves that the model could predict the probability effectively. We also estimate the predictability at a different level to study the heritability of colorectal cancer. Secondly, different types of neural networks were performed in this study. The mathematical modeling of neural networks was derived and performed, including single layer derivation, multiple layer derivation, and backpropagation theory.

The first neural network architecture we tried is the wide model. It is a single layer model which allows all the features (dense feature and sparse feature) goes in the model together. A relatively simple architecture makes the model not overfitting easily. The

parameter diagram and learning curve also prove the relatively good performance of the wide model. The second architecture we tired is the deep model. It is a popular model in recent years. However, it requires a large amount of training data to prevent overfitting since it is more complex than the wide model. So, the evaluation performance of the deep model is not satisfied. The third model we tried is the dense-sparse model. It has a better performance for the large sparse training dataset and could incorporate the interaction between the features into the model. However, it also requires a large training dataset and is easy to overfitting. The performance of the dense-sparse model was also not satisfied. The last model we tried is the pairwise model. It is inspired by the idea of predicting the similarity between images using triplet loss. It is composed of two sub neural networks and try to push the score of the positive sample and the negative sample far away using the rank loss. The wide model was used to be the sub neural network to avoid hard overfitting. Finally, the evaluation shows that the pairwise model has the best performance among these models.

By researching cancer predicting, people with high risk could screen earlier and more often and remove the tumor with surgery at an early stage. Also, people with super low risk on specific cancer could save money for screening for other cancers with higher risk.

References

1. McGuire S. World Cancer Report 2014. Geneva, Switzerland: World Health Organization, International Agency for Research on Cancer, WHO Press, 2015. Adv Nutr An Int Rev J [Internet]. 2016;7(2):418–9. Available from: http://advances.nutrition.org/cgi/doi/10.3945/an.116.012211

2. Siegel RL, Miller KD, Jemal A. Cancer statistics, 2018. CA Cancer J Clin [Internet]. 2018;68(1):7–30. Available from: http://doi.wiley.com/10.3322/caac.21442

3. The American Cancer Society medical and editorial content team. Key Statistics for Colorectal Cancer [Internet]. 2018. Available from: https://www.cancer.org/cancer/colon-rectal-cancer/about/key-statistics.html

4. American Cancer Society. Cancer Facts and Figures 2017. Genes Dev [Internet]. 2017;21(20):2525–38. Available from: https://www.cancer.org/content/dam/cancer-org/research/cancer-facts-and-statistics/annual-cancer-facts-and-figures/2017/cancer-facts-and-figures-2017.pdf

5. DeVita VT, Lawrence TS, Rosenberg SA, eds. DeVita, Hellman and R. Cancer: Principles and Practice of Oncology. 10th ed. Lippincott Williams & Wilkins;

6. Niederhuber JE, Armitage JO, Doroshow JH, Kastan MB, Tepper JE. Abeloff's Clinical Oncology: Fifth Edition. Abeloff's Clinical Oncology: Fifth Edition. 2013.

7. Steele SR, Johnson EK, Champagne B, Davis B, Lee S, Rivadeneira D, et al. Endoscopy and polyps-diagnostic and therapeutic advances in management. Vol. 19, World Journal of Gastroenterology. 2013. p. 4277–88.

8. American Cancer Society. Colorectal Cancer Facts & Figures 2017 - 2019. Atlanta. 2017;1–40.

9. Cancer Prevention Overview [Internet]. Available from: https://www.cancer.gov/about-cancer/causes-prevention/patient-prevention-overview-pdq

10. Pandey JP. Genomewide association studies and assessment of risk of disease. N Engl J Med. 2010;363(21):2076–2077; author reply 2077.

11. Bahcall O. Common variation and heritability estimates for breast, ovarian and prostate cancers [Internet]. Nature Genetics - iCOGS. 2013. Available from: http://www.nature.com/icogs/primer/common-variation-and-heritability-estimates-for-breast-ovarian-and-prostate-cancers/

12. Peters U, Hutter CM, Hsu L, Schumacher FR, Conti D V., Carlson CS, et al. Meta-analysis of new genome-wide association studies of colorectal cancer risk. Hum Genet. 2012;131(2):217–34.

13. Tomlinson IPM, Carvajal-Carmona LG, Dobbins SE, Tenesa A, Jones AM, Howarth K, et al. Multiple common susceptibility variants near BMP pathway loci GREM1, BMP4, and BMP2 explain part of the missing heritability of colorectal cancer. PLoS Genet. 2011;7(6).

14. Houlston RS, Webb E, Broderick P, Pittman AM, Di Bernardo MC, Lubbe S, et al. Meta-analysis of genome-wide association data identifies four new susceptibility loci for colorectal cancer. Nat Genet. 2008;40(12):1426–35.

15. Jaeger E, Webb E, Howarth K, Carvajal-Carmona L, Rowan A, Broderick P, et al. Common genetic variants at the CRAC1 (HMPS) locus on chromosome 15q13.3 influence colorectal cancer risk. Nat Genet. 2008;40(1):26–8.

16. Broderick P, Carvajal-Carmona L, Pittman AM, Webb E, Howarth K, Rowan A, et al. A genome-wide association study shows that common alleles of SMAD7 influence colorectal cancer risk. Nat Genet. 2007;39(11):1315–7.

17. Peters U, North KE, Sethupathy P, Buyske S, Haessler J, Jiao S, et al. A Systematic Mapping Approach of 16q12.2/FTO and BMI in More Than 20,000 African Americans Narrows in on the Underlying Functional Variation: Results from the Population Architecture using Genomics and Epidemiology (PAGE) Study. PLoS Genet. 2013;9(1).

18. Zanke BW, Greenwood CMT, Rangrej J, Kustra R, Tenesa A, Farrington SM, et al. Genome-wide association scan identifies a colorectal cancer susceptibility locus on chromosome 8q24. Nat Genet [Internet]. 2007;39(8):989–94. Available from: http://www.ncbi.nlm.nih.gov/entrez/query.fcgi?cmd=Retrieve&db=PubMed&dopt=Citation&list_uids=17618283%5Cnhttp://www.ncbi.nlm.nih.gov/pubmed/17618283

19. Tomlinson IP, Webb E, Carvajal-Carmona L, Broderick P, Howarth K, Pittman AM, et al. A genome-wide association study identifies colorectal cancer susceptibility loci on chromosomes 10p14 and 8q23.3. Nat Genet [Internet]. 2008;40(5):623–30. Available from: http://www.nature.com/ng/journal/v40/n5/pdf/ng.111.pdf

20. Dunlop MG, Dobbins SE, Farrington SM, Jones AM, Palles C, Whiffin N, et al. Common variation near CDKN1A, POLD3 and SHROOM2 influences colorectal cancer risk. Nat Genet. 2012;44(7):770–6.

21. Jia W-H, Zhang B, Matsuo K, Shin A, Xiang Y-B, Jee SH, et al. Genome-wide association analyses in East Asians identify new susceptibility loci for colorectal cancer. Nat Genet [Internet]. 2013;45(2):191–6. Available from: http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=3679924&tool=pmcentrez&rendertype=abstract

22. Gong J, Hutter C, Baron JA, Berndt S, Caan B, Campbell PT, et al. A pooled analysis of smoking and colorectal cancer: Timing of exposure and interactions with environmental factors. Cancer Epidemiol Biomarkers Prev. 2012;21(11):1974–85.

23. Houlston RS, Cheadle J, Dobbins SE, Tenesa A, Jones AM, Howarth K, et al. Meta-analysis of three genome-wide association studies identifies susceptibility loci for colorectal cancer at 1q41, 3q26.2, 12q13.13 and 20q13.33. Nat Genet. 2010;42(11):973–7.

24. Peters U, Jiao S, Schumacher FR, Hutter CM, Aragaki AK, Baron JA, et al. Identification of genetic susceptibility loci for colorectal tumors in a genome-wide meta-analysis. Gastroenterology. 2013;144(4).

25. Jiao S, Peters U, Berndt S, Brenner H, Butterbach K, Caan BJ, et al. Estimating the heritability of colorectal cancer. Hum Mol Genet. 2014;23(14):3898–905.

26. O'Brien JM. Environmental and heritable factors in the causation of cancer analyses of cohorts of twins from Sweden, Denmark, and Finland,. Vol. 45, Survey of Ophthalmology. 2000. p. 167–8.

27. Redon R, Ishikawa S, Fitch KR, Feuk L, Perry GH, Andrews TD, et al. Global variation in copy number in the human genome. Nature. 2006;444(7118):444–54.

28. Bishop CM. Pattern Recognition and Machine Learning [Internet]. Vol. 4, Pattern Recognition. 2006. 738 p. Available from: http://www.library.wisc.edu/selectedtocs/bg0137.pdf

29. Cossock D. Statistical Analysis of Bayes Optimal Subset Ranking.

30. Schwartz WB, Patil RS, Szolovits P. Artificial Intelligence in Medicine. N Engl J Med. Massachusetts Medical Society ; 1987 Mar;316(11):685–8.

31. Gorry GA, Barnett GO. Sequential Diagnosis by Computer. JAMA J Am Med Assoc. American Medical Association; 1968 Sep;205(12):849.

32. Monroe D. Deep learning takes on translation. Commun ACM. 2017 May;60(6):12–4.

33. Halevy A, Norvig P, Pereira F. The Unreasonable Effectiveness of Data. IEEE Intell Syst. 2009 Mar;24(2):8–12.

34. Peterson W, Birdsall T, Fox W. The theory of signal detectability. Trans IRE Prof Gr Inf Theory. 1954 Sep;4(4):171–212.

35. Thompson IM, Ankerst DP, Chi C, Lucia MS, Goodman PJ, Crowley JJ, et al. Operating Characteristics of Prostate-Specific Antigen in Men With an Initial PSA Level of 3.0 ng/mL or Lower. JAMA. American Medical Association; 2005 Jul;294(1):66.

36. Reichlin T, Hochholzer W, Bassetti S, Steuer S, Stelzig C, Hartwiger S, et al. Early Diagnosis of Myocardial Infarction with Sensitive Cardiac Troponin Assays. N Engl J Med. Massachusetts Medical Society ; 2009 Aug;361(9):858–67.

37. Norton ME, Jacobsson B, Swamy GK, Laurent LC, Ranzini AC, Brar H, et al. Cell-free DNA Analysis for Noninvasive Examination of Trisomy. N Engl J Med. Massachusetts Medical Society; 2015 Apr;372(17):1589–97.

38. Tankova T, Chakarova N, Dakovska L, Atanassova I. Assessment of HbA1c as a diagnostic tool in diabetes and prediabetes. Acta Diabetol. 2012 Oct;49(5):371–8.

39. Poldervaart JM, Langedijk M, Backus BE, Dekker IMC, Six AJ, Doevendans PA, et al. Comparison of the GRACE, HEART and TIMI score to predict major adverse cardiac events in chest pain patients at the emergency department. Int J Cardiol. Elsevier; 2017 Jan;227:656–61.

40. Tanaka F, Yoneda K, Kondo N, Hashimoto M, Takuwa T, Matsumoto S, et al. Circulating tumor cell as a diagnostic marker in primary lung cancer. Clin Cancer Res. American Association for Cancer Research; 2009 Nov;15(22):6980–6.

41. Lloyd S. Least squares quantization in PCM. IEEE Trans Inf Theory. 1982 Mar;28(2):129–37.

42. Srivastava N, Hinton G, Krizhevsky A, Sutskever I, Salakhutdinov R. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. J Mach Learn Res. 2014;15:1929–58.

43. Ahlqvist E, Storm P, Käräjämäki A, Martinell M, Dorkhan M, Carlsson A, et al. Novel subgroups of adult-onset diabetes and their association with outcomes: a data-driven cluster analysis of six variables. lancet Diabetes Endocrinol. Elsevier; 2018 May;6(5):361–9.

44. Richard O. Duda, Peter E. Hart DGS. Pattern classification. 2nd ed. Wiley; 2001.

45. Clarke GM, Anderson CA, Pettersson FH, Cardon LR, Andrew P. Basic statistical analysis in genetic case-control studies. 2011;6(2):121–33.

46. Institute NHGR. Genome-Wide Association Studies [Internet]. 2015. Available from: https://www.genome.gov/20019523/genomewide-association-studies-fact-sheet/

47. Trait: colorectal cancer [Internet]. GWAS Catalog. Available from: https://www.ebi.ac.uk/gwas/efotraits/EFO_0005842

48. ISB Cancer Genomics Cloud 1.0.0 documentation [Internet]. Available from: http://isb-cancer-genomics-cloud.readthedocs.io/en/latest/sections/data/data2/data_in_BQ.html#

49. Friedman JH. Greedy function aproximation: A gradient boosting machine. Ann Stat [Internet]. 2001;29(5):1189–232. Available from: http://www.jstor.org/stable/2699986%0Ahttp://about.jstor.org/terms

50. Click C, Malohlava M, Candel A, Roark H, Parmar V. Gradient Boosting Machine with H2O. Https://WwwH2OAi/Resources/ [Internet]. 2017;(6):30. Available from: http://h2o-release.s3.amazonaws.com/h2o/master/3805/docs-website/h2o-docs/booklets/GBMBooklet.pdf

51. Charmaz K. Additive logistic regression: a statistical view of boosting. Int J Qual Methods [Internet]. 2017;16(1):160940691771935. Available from: http://journals.sagepub.com/doi/10.1177/1609406917719350

52. Wpengine. Building a Distributed GBM on H2O [Internet]. h2o.ai. 2013. Available from: https://blog.h2o.ai/2013/10/building-distributed-gbm-h2o/

53. Majnik M, Bosni Z. ROC analysis of classifiers in machine learning: A survey. Vol. 17, Intelligent Data Analysis. 2013. p. 531–58.

54. Kim M, Kim S-H. Empirical prediction of genomic susceptibilities for multiple cancer classes. Proc Natl Acad Sci [Internet]. 2014;111(5):1921–6. Available from: http://www.pnas.org/lookup/doi/10.1073/pnas.1318383110

55. Wray NR, Visscher PM. Estimating trait heritability. Nat Educ. 2008;1(1):29.

56. Diebold FX, Kilian L. Measuring predictability: Theory and macroeconomic applications. J Appl Econom. 2001;16(6):657–69.

57. DeepAI. Neural Network [Internet]. Available from: https://deepai.org/machine-learning-glossary-and-terms/neural-network

58. Hagan, M. T., Demuth, H. B., Beale, M. H., & De Jesús O. Neural Network Design. Vol. 20, Boston: Pws Pub. 1996.

59. Nair V, Hinton G. Rectified Linear Units Improve Restricted Boltzmann Machines. In: Proceedings of the 27th International Conference on Machine Learning. 2010.

60. Maas AL, Hannun AY, Ng AY. Rectifier nonlinearities improve neural network acoustic models. In: ICML ’13. 2013.

61. Xu B, Wang N, Chen T, Li M. Empirical Evaluation of Rectified Activations in Convolutional Network. 2015; Available from: http://arxiv.org/abs/1505.00853

62. Deng Z. Stochastic Area Pooling for Generic Convolutional Neural Network. CLR2016. 2016;

63. He, K., Zhang, X., Ren, S., & Sun J. Deep Residual Learning for Image Recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition. 2016.

64. Anthony M, Mathematics A, Science P, Bartlett P, Fellow S, Sciences I. Neural Network Learning : Theoretical Foundations.

65. Olegalexandrov. Wikimedia Commons [Internet]. 2012. Available from: https://commons.wikimedia.org/wiki/File:Gradient_descent.svg

66. Ciregan D, Meier U, Schmidhuber J. Multi-column deep neural networks for image classification. In: Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition. 2012.

67. TensorFlow. TensorBoard: Visualizing Learning. TensorFlow. 2017.

68. Bengio Y. Learning Deep Architectures for AI. Found Trends® Mach Learn. 2009;

69. Rendle S. Factorization machines. Proc - IEEE Int Conf Data Mining, ICDM. 2010;995–1000.

70. Guo H, Tang R, Ye Y, Li Z, He X. DeepFM: A factorization-machine based neural network for CTR prediction. IJCAI Int Jt Conf Artif Intell. 2017;1725–31.

71. Hadsell R, Chopra S, LeCun Y. Dimensionality reduction by learning an invariant mapping. In: Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition. 2006.

72. Taylor GW, Spiro I, Bregler C, Fergus R. Learning invariance through imitation. In: Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition. 2011.

73. Lowe DG. Object recognition from local scale-invariant features. In 2008.

74. Dalal N, Triggs B. Histograms of oriented gradients for human detection. In: Proceedings - 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, CVPR 2005. 2005.

75. Wang J, Song Y, Leung T, Rosenberg C, Wang J, Philbin J, et al. Learning fine-grained image similarity with deep ranking. In: Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition. 2014.