EE359 Data Mining

Project Report

Xianze Wu*
515030910573
Shanghai Jiao Tong University
800 Dongchuan Road,
Minhang District, Shanghai
xzwu20@163.com

Mingcheng Chen†
515030910568
Shanghai Jiao Tong University
800 Dongchuan Road,
Minhang District, Shanghai
cmc_iris@163.com

ABSTRACT

This report is the project report for gene chips analysis in EE359. In this project, we firstly scrubbed the whole data set to selected data with high quality and complete labels, after that we applyed Principal Component Analysisand(PCA) and autoencoder on the dataset to reduce the dimension of data features. Then we tried both different classical machine learning methods, using k-NN method, Logistic Regression, Support vector machine(SVM), Decision Tree, and deep learning method on the multiple classification task and binary classification task respectively. In this report the first part introduce the basic theory of our models. Then the following part introduce the detail of our experiments. Experiment result and corresponding discussion part given at the end of the experiments and results part, we will display our experiment result and do comparison and analysis on the results.

Keywords

Gene chips analysis; Classification; Machine learning; Deep learning

1. INTRODUCTION

Human gene is the complete set of nucleic acid genomic sequence (DNA or RNA) which is the basic physical and functional unit of heredity. A gene can be directly functional or be the intermediate template for a protein that performs a specific function[5]. Most biological traits are under the influence of poly-genes as well as gene-environment interactions[11]. In particular, risk to a range of diseases is directly associated with corresponding genes or their mutation. Therefore, medical diagnostics to these diseases can be accomplished with gene analysis, which promises high accuracy[10][2].

The advent of gene chips[14] furthermore allow humans determine the diseases and learning the gene expression states by modern machine learning approaches, which can handle up to 500 million base pairs of DNA with thousands of genes. Benefit from that, the popular data mining methods can be used to solve many biogenome task, which can in turn improve the human knowledge towards AI algorithm and lead an era of big data mining.

Some traditional, classical classification models have certified their accuracy and robustness in supervised learning. Especially in classification tasks which aim to classify examples into given set of categories. Classical machine learning methods like Logistic Regression, SVM, k-NN and Dicision Tree can interpret a specific task into statistical classification. All of these methods have the possibility of achieving great performance dealing with gene chips analysis tasks.

Rapid development of deep neural network, also called deep learning or hierarchical learning, has achieved great performance on applications in Computer Vision and Nature Language Processing fields. And human see the great potential of applying it on gene chips data mining and helps determine gene diseases.

In this project, raw dataset[1] of some genes and their feature observation are given. The data shows observation results of gene chips microarray from several kinds of cells, some of which are healthy cells and others are cancer cells or pathological cells. Our task is to carry out *multi-classification* and *binary-classification* on given data set. We firstly reduce the dimension of data features by PCA methods. Then we do the classification task using both traditional classical methods, Logistic regression, SVM, Dicision Tree and k-Nearest Neigbour contained, and Deep Learning based neural network framework.

Our code[2] is written in `Python3.5` under the `PyTorch` framework, a deep learning framework for fast, flexible experimentation, and `scikit-learn` machine learning library, a simple and efficient tools package for data mining and data analysis in `Python`. Experiments were carried out under environment of os `Ubuntu 16.04 LTS`, corresponding thoughts and discussion are attached to the experiment results.

The following part in this report will introduce the basic theory of our models. Then at the third part, we will in-

*Xianze Wu takes responsibility for deep learning method in this project as the team leader.

†Mingcheng Chen is responsible for classical method.

[1]Dataset can be download from website E-TABM-185.

[2]Our code is available at `Github`.

troduce details of our experiments and show the experiment results. The result comparison, analysis and discussion are also in the third part.

2. METHODS

In this part, we will mainly give a description of machine learning methods used in our experiment and how they are realized. In general, it could be divided into three sections-dimension reduction, classical methods and deep learning based methods.

2.1 Dimension Reduction

Since the given dataset has dimension of 22283 while only 3570 valid items, which is a problem with large p and small n. Therefore, dimension reduction process before applying machine learning methods is necessary. To achieve the aim of dimension reduction, we chose the mainstream method PCA and autoencoder.

Here we will give a brief introduction of the basic theory of PCA and autoencoder. How these methods' functionalities will be discussed in the experiment and result part in detail.

2.1.1 PCA Method

Principal Components analysis(PCA) [13] aims to perform a linear mapping of the data to a lower-dimensional space, in such a way that the variance of the data in the low-dimensional representation is maximized. Suppose there is a given data matrix X with size $n \times p$, where n is the quantity of sampling and p is the dimension of feature. Usually, the number of dimension after PCA is smaller than n. Some important steps of PCA algorithm are listed as follows

1) Normalize the data with the empirical mean vector and let it be centered. Approximate the empirical mean vector u from known samples by using the averaging expression Eq(1),

$$u = \frac{1}{n}\sum_{i=1}^{n} x_i \tag{1}$$

then centered the data by Eq(2);

$$x = x - u \tag{2}$$

2) Calculate the covariance matrix C from the samples with

$$C = \frac{X^T X}{n-1} \tag{3}$$

3) Compute the eigenvectors and eigenvalues of the covariance matrix C, since C is a symmetric matrix and thus can be diagonalized

$$V^T C V = D \tag{4}$$

where the matrix V consists of eigenvectors which diagonalized the covariance matrix C, and D is the diagonal matrix of eigenvalues of C.

4) Rearrange the eigenvectors and eigenvalues in a way that the columns of the eigenvector matrix V and eigenvalue matrix D are sorted in order of decreasing eigenvalue. The eigenvalues represent the distribution of the source data's energy among each of the eigenvectors, where the eigenvectors form a basis for the data.

5) Select the first k columns of V as the $p \times k$ matrix W, and the criterion should be followed Eq(5),

$$\frac{\sum_{i=1}^{k} D_{ii}}{\sum_{i=1}^{p} D_{ii}} \geq \mathit{Threshold} \tag{5}$$

where *Threshold* is the desired proportion of cumulative energy. In this project, we tried different threshold, $\mathit{Threshold} = 95\%, 90\%, 85\%$, to get a proper final p.

6) Project the z-scores of the data onto the new basis

$$t_i = W^T(x_i - u) \tag{6}$$

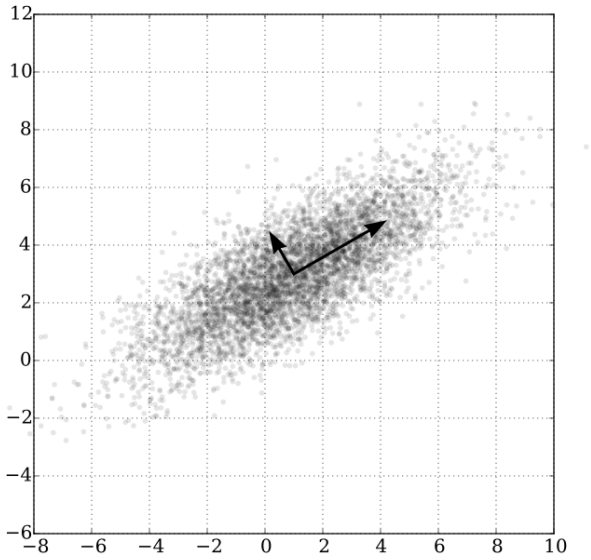

Figure 1: PCA of a Multivariate Gaussian Distribution. (from Wikipedia)

In such a set of steps, PCA makes the feature has large variance while the noise has low variance. After PCA, we successfully reduce the correlation of features, which considerably accelerates training in following classification stages. Moreover, centered feature vector helps a lot for training without extra preprocessing.

2.1.2 Autoencoder

There is another method using autoencoder neural network to achieve dimension reduction. Different from PCA using traditional mathematical operation to project the high-dimension original features onto lower dimension one, an autoencoder using neural network and is an unsupervised learning algorithm that applies back propagation, setting the target values to be equal to the inputs. The structure of autoencoder is shown as Fig2.

Assume that input layer $L1$ here is a set of examples $X = [x_1, x_2, x_3, \ldots]$, we hope $X = H_{W,b}(X)$. Then the middle layer $L3$ with the lowest dimension output z is the encoder of the raw data X. The key difference between autoencoder and PCA is that, this output in an autoencoder

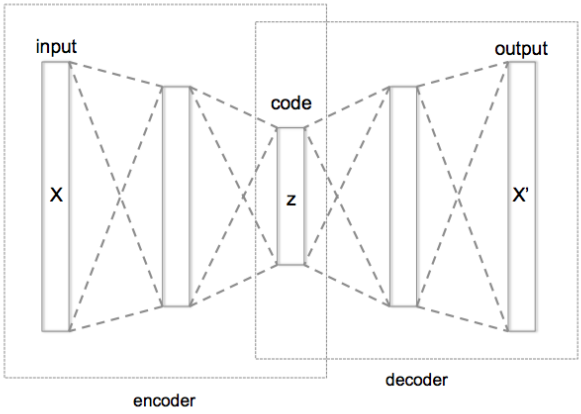

Figure 2: Autoencoder Structure.

is a nonlinear dimension reduction, and it may overcome the weakness of PCA that it is only a linear dimension reduction algorithm.

2.2 Classical Methods

The classical methods we use containing (1)logistic regression, (2)multi-class support bector machine, (3)dicision tree method and (4)the k-nearest neighbour algorithm.

2.2.1 *Logistic Regression*

Logistic regression model is a classical supervised linear classifier. The linear classification approach has two major components: a *score function* that maps the raw data to class scores, and a *loss function* that quantifies the gap between the predicted scores and the ground truth labels.

What logistic regression do is, firstly calculates the boundary among different classes, and then it will predict the possibility of test data class based on the calculated data boundary.

Let x denotes the feature vector of given training dataset, and Y denotes corresponding label of training dataset, then the calculated class possibility is defined as Eq(7)

$$P(Y = y_i|x_i) = \frac{e^{\omega x_i + b}}{1 + \sum_i e^{\omega x_i + b}} \tag{7}$$

where ω denotes the regression weights and b denotes the regression bias. Based on the labels training data, using Maximum Likelihood Estimation(MLE) method we can determine the value of ω. Let $h_\omega(x_i)$ denoting the possibilities, the log likelihood function is defined as Eq(8),

$$L(\omega) = \sum_i^N [y_i \log h_\omega(x_i) + (1 - y_i) \log(1 - h_\omega(x_i))] \tag{8}$$

There are several gradient-based optimizer can be applied to determine the value of ω and b. In our project, we test logistic regression method on both multi-classification and binary-classfication tasks, compare model performance with different optimizer and regularization methods.

2.2.2 *Softmax Classifier*

Softmax Classfier[7] is the generalization of *logistic regression* model which we have introduced above, or in other words, *softmax regression* is a direct promotion of logistic regression in multiple classifications. For this reason, *Softmax Classifier* is usually called *Multinomial Logistic Regression* model.

Differ from *logistic regression*, *softmax classifier* models probabilities with the **softmax function**, which is defined by

$$P(y = i|x, \theta) = \frac{e^{\theta_i^T x}}{\sum_j^K e^{\theta_j^T x}} \tag{9}$$

and the decision function for softmax is given by

$$y^* = \arg\max_i P(y = i|x, \theta) \tag{10}$$

while the corresponding loss function is

$$J(\theta) = -\frac{1}{N} \sum_i^N \sum_j^K 1[y_i = j] \log \frac{e^{\theta_i^T x}}{\sum e^{\theta_k^T x}} \tag{11}$$

Similar to *logistic regression*, we can solve *softmax regression* problem by methods based on gradient descent or other high-order methods, and we will not repeat here.

2.2.3 *Support Vector Machine*

Support Vector Machine (SVM) is a supervised learning model proposed by [3]. The idea of SVM is to find a hyper plane n the feature space so that it can separate differnet samples into different categories.

SVM is also a linear classifier model. However, defferent from other linear model like *logistic regression*, SVM aims to maximize the *margin* between two different categories' samples and the samples on the boundary are so-called *support vectors*.

However, we often meet the situation that the samples are not linear separable in real operation, so trick named *kernel method* is introduced [4]. Kernel trick project samples with finite dimensions to a high dimension or even infinite dimension space implicitly, such project pulls samples apart so that they can be separated in the project space with hyper plane. If we denote such a project by $\phi(\mathrm{x}_i)$, where x_i is the original vector, we have:

$$\mathcal{K}(\mathrm{x}_i, \mathrm{x}_j) = \langle \phi(\mathrm{x}_i), \phi(\mathrm{x}_j) \rangle = \phi(\mathrm{x}_i)^T \phi(\mathrm{x}_j) \tag{12}$$

Since the mapping from the input space to the feature space will cause the exponential growth of the dimension, the operation of the inner product $\phi(\mathrm{x}_i) \cdot \phi(\mathrm{x}_j)$ in the above constraint problem will be too large to bear, so usually we will construct a kernel function as Eq(13)

$$\mathcal{K}(\mathrm{x}_i, \mathrm{x}_j) = \phi(\mathrm{x}_i) \cdot \phi(\mathrm{x}_j) \tag{13}$$

by this way, the computation in the feature space is avoided, and the inner product operation in the feature space can be performed only in the input space. Some of the common kernel functions are listed as follows:

1) linear kernel:

$$\mathcal{K}(\mathrm{x}_i, \mathrm{x}_j) = \mathrm{x}_i^T \mathrm{x}_j \tag{14}$$

2) polynomial kernel:

$$\mathcal{K}(\mathrm{x}_i, \mathrm{x}_j) = (\gamma \mathrm{x}_i^T \mathrm{x}_j + r)^d, d > 1 \tag{15}$$

3) RBF (Gaussian) kernel:

$$\mathcal{K}(\mathrm{x}_i, \mathrm{x}_j) = \exp(-\gamma||\mathrm{x}_i - \mathrm{x}_j||^2), \gamma > 0 \quad (16)$$

4) sigmoid kernel:

$$\mathcal{K}(\mathrm{x}_i, \mathrm{x}_j) = \tanh(\gamma \mathrm{x}_i^T \mathrm{x}_j + r), \gamma > 0, r < 0 \quad (17)$$

Hinge Loss is a commonly used loss in SVM. Denote fixed margin as Δ, and use s to represent the class scores returned by *score function*. With the score for the j_{th} class is the j_{th} element: $s_j = f(x_i, W, b)_j$, the multiclass SVM loss for the i_{th} sample is then formalized as foolowed:

$$L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i}) + \Delta \quad (18)$$

With the SVM loss function, the threshold at zero $\max(0, -)$ function is often called the *hinge loss*. And the so-called *squared hinge loss SVM* (i.e. L2-SVM) uses the form $\max(0, -)^2$ that penalizes violated margins more strongly. The unsquared version is more standard, but in some datasets the squared hinge loss can work better. This can be determined during cross-validation. [8]

2.2.4 K-Nearest Neighbor

K-Nearest Neighbour (k-NN) is a non-parametric method used for pattern recognition and classification. The idea of k-NN is described as follows: Given a sample x, the classifier will compute the distance between x and other samples in dataset X. Then it will find the top k closest samples X'. In k-NN point of view, the most common class in X' will be the class of x.

The function that measured the distance mentioned above called a *distance function*. Although such function can be defined in many ways, the Euclidean metric is the most common practice which maps the straight-line distance between two points in Euclidean space. Suppose two feature vectors x_1, x_2 are given, both of which have dimension p, the L1 distance is defined by

$$d_1(x_1, x_2) = \sum_p |x_1^p - x_2^p| \quad (19)$$

Another common choice of distance function is L2 distance, which has the geometric interpretation of computing the Euclidean distance between two vectors. L2 distance is given by the expression:

$$d_1(x_1, x_2) = \sqrt{\sum_p (x_1^p - x_2^p)^2} \quad (20)$$

K-NN model has its own incomparable advantages. It is very simple to implement and understand. Moreover, the classifier takes no need to train, since all it requires is to store and index the dataset.

We apply k-NN classifier on both the multi-label prediction task and binary-label precdiction process, and we find that k-NN leads us care more about test efficiency but not training efficiency, that is one of its shortcomings.

2.3 Deep Learning Based Methods

The concept of Neural Networks (NN)[1] is originally inspired by the goal of modeling biological neural systems, but has since diverged and become a matter of engineering and achieving good results in Machine Learning tasks.

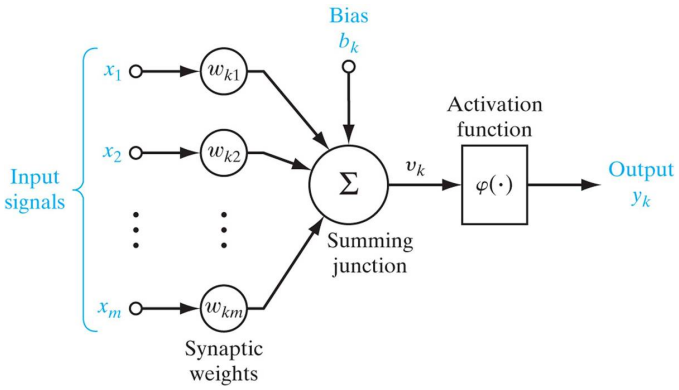

Figure 3: Artificial Neural Network Neuron Schematic.

2.3.1 Weight initialization

Initial neural network properly is important. A simple way is to set all weights to 0 at first, however, it results in all neural unit compute the same gradients during backpropagation. An alternative way is normal distribution. However, as all units are initialized with a number in $[0, 1]$, variance of the network will grow with the number of inputs. Ultimately, we use xavier normalization[6] to avoid theses problems.

$$\omega \sim N(0, \sqrt{\frac{2}{fan_{in} + fan_{out}}}) \quad (21)$$

where fin_{in} is input dimension of the network and fan_{out} is output dimension of the network.

2.3.2 Regularization

Regularization is the key method to prevent overfitting [8]. Several ways of regularization are listed as follows:

1) *L2 Regularization* is a method of adding a regularization term after the cost function:

$$C = C_0 + \frac{\lambda}{2n} \sum_w w^2 \quad (22)$$

C_0 in the equation represents the original cost function, the latter term is L2 regularization term, its working machanism is: using the sum of the squares of all parameters w, divided by the sample size of the training set n. λ is the regular term coefficient, weighing the proportion of the regular term and the C_0 term. Notice that, the coefficient $1/2$ is unique, this term of constant is designed for rounding up the results of following derivation.

2) *L1 Regularization* adds a regularization term after original cost function, which is the sum of absolute value of all weight w, and times λ/n as shown in Eq(23)

$$C = C_0 + \frac{\lambda}{n} \sum_w |w|. \quad (23)$$

In our experiment, we use L2 regularization in each fullt-connected layer but not L1 regularization for a better efficiency.

3) *Dropout*[12]: L1 and L2 regularization regularize the data by modifying the cost function, while *Dropout*

achieve the aim of regularization by modifying the framework itself of neural network. It is a useful trick during the training of deep neural network to prevent overfitting. The process of dropout shown in following figures,

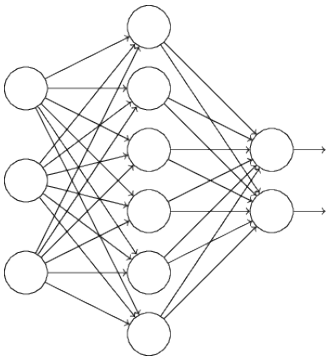

Figure 4: Neural Network Before Dropout.

Suppose we want to train the network shown above. At the beginning of training, we delete p percentage of the hidden layer units, so we get the network framework as follows:

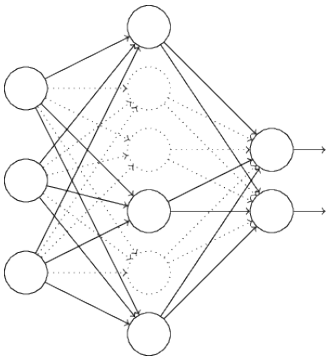

Figure 5: Neural Network After Dropout.

Keeping the input and output layers unchanged, updating the weights in neural network after dropout according to BP algorithm (do not update the weights of units be dropped in the first step). So that each epoch give out a result using p percentage of the neurons, synthesis all the results got by large quantities of epochs, the final result should be with less wrong classification result influence, thus prevent overfitting.

We carried out both Dropout trick and L2 regularization to control overfitting, the effect is well and show in experiment and results part.

3. EXPERIMENTS AND RESULTS

3.1 Tasks and Data Processing

After analyzing the GeneChip dataset, we choose the *MaterialType* and *DiseaseState* item as classification labels. There are only two type of material in the dataset, so classification on *MaterialType* is a **binary-classification** problem. However, not all samples in the dataset has *d*isease state label. Besides, some disease state rarely appear. Hence, we preprocess the dataset to remove those samples without disease state label or corresponding label appears less than 10 times. The processed dataset consists of 3530 samples with 2 material types and 79 disease states, so classification on *DiseaseState* is a **multi-classification** problem. Then, we apply pca to the processed dataset and dividing the dataset into training and testing set, with ratio 8:2. There are 2824 samples in training set and 706 samples in testing set.

3.2 Principal Components Analysis

We apply principal components analysis method to reduce the dimension of samples in the dataset. We determine the dimension of processed data by setting the desired proportion of cumulative energy(denoted as $Threshold$ in following part), as described in section 2.1. To evaluate the quality of samples with different dimensions, we test them in the multi-classification task, with adopting a logistic regression model. Results are as follows. Note that L2 regularization is applied and l2 factor is 1.

Table 1: Evaluation result for samples with different dimension

$Threshold$	Dimension	Accuracy
0.6	13	0.25
0.65	19	0.34
0.7	29	0.4143
0.75	48	0.5574
0.8	87	0.6235
0.85	183	0.7147
0.9	435	0.7567
0.95	1048	0.7116

Hence, we set the $Threshold$ as 0.9 finally, and the feature is reduced to 453 dimensions.

3.3 Evaluation Metrics

For classification task, accuracy is a direct and intuitive evaluation metric. However, sometimes it is not reliable. For example, in the dataset, there are about 80% samples has the disease state "organism part", which means the model can achieve 80% accuracy if it predicts disease state of all samples as "organism part". To handle this problem, we apply F1-score. In summary, f1-score is defined as

$$f1\ score = 2 * \frac{precision * recall}{precision + recall} \tag{24}$$

$precision$ is defined as

$$precision = \frac{True\ Positive}{True\ Positive + False\ Positive} \tag{25}$$

and $recall$ is defined as

$$recall = \frac{True\ Positive}{True\ Positive + False\ Negative} \tag{26}$$

. Informally, the former evaluates the result from predicted labels' perspective and the latter evaluates the result from targets' perspective.

Besides, both micro and macro F1-score are applied. Micro F1-score is calculated by global *True Positive*, *False Positive* and *False Negative*. Macro F1-score is unweight mean

of F1-score for each label. We use accuracy and micro F1-score to evaluate the performance on binary-classification. Accuracy and macro F1-score are adopted to evaluate the performance on multi-class classification.

3.4 Linear regression with softmax loss

In this part, we apply a linear regression model with softmax loss. L2 regularization is applied, and we try different regularization factor λ. Results are shown in table 3.4.

Table 2: Performances of linear regression with softmax loss

Table 3: Performances on binary-classification

	$\lambda = 0.01$	$\lambda = 0.1$	$\lambda = 1$	$\lambda = 10$	$\lambda = 100$
accuracy	0.978	0.978	0.981	0.981	0.977
micro f1	0.893	0.893	0.889	0.889	0.866

Table 4: Performances on multi-classification

	$\lambda = 0.01$	$\lambda = 0.1$	$\lambda = 1$	$\lambda = 10$	$\lambda = 100$
accuracy	0.862	0.862	0.843	0.839	0.839
macro f1	0.818	0.814	0.799	0.793	0.797

In general, as λ increasing, regularization strength becomes heavier and the model perform worse.

3.5 Linear regression with hinge loss

In this part, we apply a linear regression model with hinge loss. We try different C. For binary-classification, algorithms with various C all achieve 0.9579 and f1 score 0.8595. For multi-classification, results are shown in table 3.5.

Table 5: Performances on multi-classification

	$C = 0.01$	$C = 0.1$	$C = 1$	$C = 10$	$C = 100$
accuracy	0.796	0.813	0.806	0.816	0.806
macro f1	0.660	0.768	0.764	0.772	0.760

3.6 K-nearest Neighbors

In this part, we apply the K-nn algorithm with various K. For binary-classification task, to our surprise, models with k = 1, 3 and 5 all achieve accuracy 1 and f1 score 1. For multi-classification task, results are as follows.

Table 6:

K	Accuracy	macro F1
1	0.8328	0.7933
3	0.8413	0.7806
5	0.8456	0.758

3.7 Deep Learning Methods

Our neural network consists of 3 layers, they have 128, 256 and 128 neural units respectively. A linear classifier with 79 neural units is connected to the last layer of the network. Cross-entropy loss with L2 regularization is adopted. Besides, we use dropout before the network and between each layer in the network. Training is stopped if no improvement is observed in the training set for 100 epochs. For parameters update, we use adam optimizer[9] empirically and initial learning rate are set to 0.005.

Finally, we achieve accuracy 0.9957 and micro F1 0.9840 on the binary-classification task, and accuracy 0.8668 and macro F1 0.8424 on the multi-classification task.

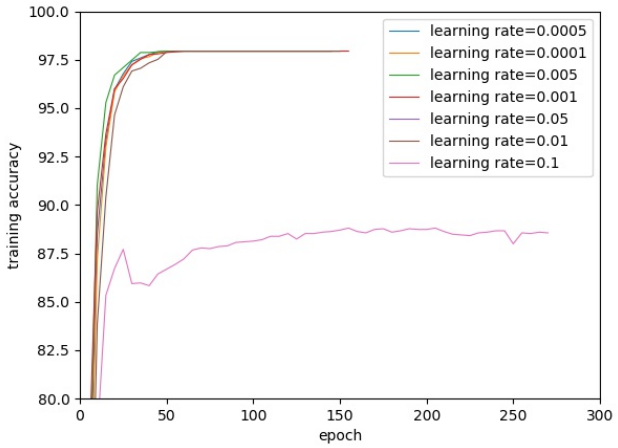

Figure 6: training accuracy curve for different learning rate

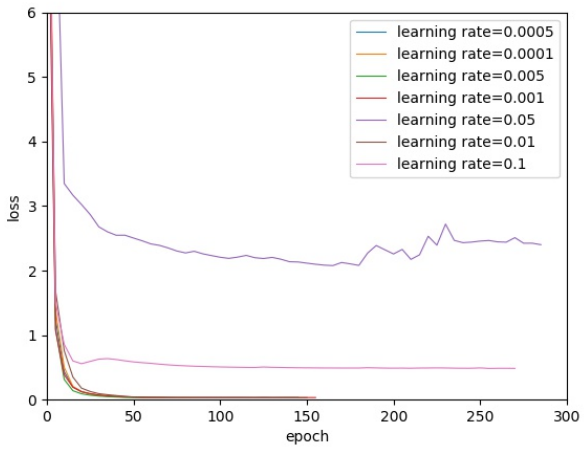

Figure 7: training loss curve for different learning rate

4. CONCLUSIONS AND THOUGHTS

4.1 PCA Parameter Choosing

The original data is a large p and small n problem, so we carried out PCA to do dimension reduction work. We carried out *baseline* multi-label prediction task over different preprocessed datasets derived under PCA approaches with *threshold* range from 60% to 95%, (actually 60%, 65%, 70%, 75%, 80%, 85%, 90%, 95%), and the dimension of preprocessed datasets range from 13 to 1048, (actually 13, 19, 29, 48, 87, 183, 435, 1048). And we find that, with the increasing of threshold, the accuracy is fisrtly goes up and then

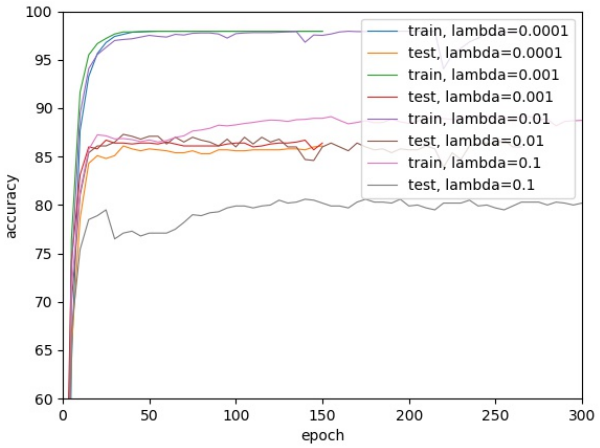

Figure 8: training and testing accuracy curve associated with l2 regularization penalty λ

drops instead. As shown in Fig.9, we get the best accuracy when we use the threshold of value 90%.

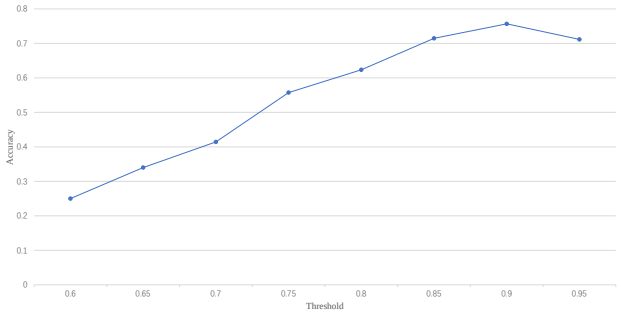

Figure 9: Prediction Accuracy Fluctuates with PCA Threshold.

When we conduct this set of experiment (0.9 threshold, 453 data dimensions, 0.7567 baseline accuracy), we reached the best accuracy of 75.67% in the testing set. We have reason to believe, for given raw dataset, the complexity of dataset and the remaining information of data reached a balance point at the point of 90% PCA threshold, and we continue our experiment with the *Threshold* set as 0.9 finally, and the featureis reduced from original 22283 dimensions to 453 dimensions.

4.2 Selecting Optimizer And Parameters For Network

Empirically, we select Adam optimizer at first. Further, we also test SGD optimizer and Adadelta optimizer. We find that Adam optimizer achieve the best performance, and Adadelta performs slightly worse than it and SGD performs the worst, with a clear gap with Adam and Adadelta. Besides, when set with same learning rate, Adadelta converges obviously slower than Adam.

Finally, we set learning rate as 0.005 and penalty term of L2 regularization as 0.0001. We try learning rate in $\{10^{-5}, 5 * 10^{-4}, 10^{-4}, ..., 0.05, 0.1\}$ and penalty term in $\{10^{-5}, 10^{-4}, ..., 0.1\}$, we also try to reduce neural units number of each layer or layer number in the network. Many attempts achieve an accuracy about 86% on multi-classification task, however we fail to get higher results.

4.3 Discussion on Dataset

Our model can achieve about 98% accuracy on multi-classification task in training set, while it only achieves 86% in testing set. At first, we hesitate that the model suffers from over fitting. However, as shown in 8, the case that training accuracy is increasing while testing accuracy is decreasing, which indicates overfitting, doesn't appear. According to these phenomenon, we think that maybe the dataset is not enough for training our neural network model. Besides, this small dataset masks some disadvantages of classical methods.

4.4 Comparison between Deep Learning and Classical Methods

Our experiments realized both classical machine learning methods and deep learning based methods. These two kinds of classification methods differ in many aspects.

1) **The design and framework**: When we use classical machine learning methods, such as K-NN, SVM and Logistic Regression, we mainly pay our attention on *score function*, *loss function* and how to use algorithm to solve the problem. Take an example, SVM try to seperate the dataset using a hyper plane, so it define the kernel method and use the kernel functions to project data into high-dimension abstract space for data point seperating. When it comes to deep learning method, the details of task is not that specific, or in other word, we care more about the structure and framework of our model, just like how to build a resonable structure or how to choose a proper set of parameters.

2) **Sensitivity to data volume**: During our experiments, we found that when the data volume is not large enough, it maybe not a good choice to use deep learning methods. Deep learning models need adjusting a lot of additional parameter than classical machine learning methods like LR and SVM. Therefore, it need more data to finish the training process and ensure the model convergency without overfitting or underfitting. As statistical result shows, the valid items in our given E-TABM-185 gene chips dataset are only 3530, which is much not enough comparing to the original 22283 dimensions. Thus, we have to do a lot of additional work like applying PCA dimension reduction, applying L2 regularization, using batch normalization, doing oversample and undersample and so on to try our best improving deep learning model performance. In contract, classical machine learning methods require less on data volume, when dealing with the same dataset, most of classical methods work well, especially 2-way k-NN reach even 1.0 accuracy and f1 score.

3) **Scalability and optimisation**: By the experiments and comparing the results, we get a conclusion that

deep learning method has both higher scalability and optimisation than classical machine learning methods. During the experiments of classical machine learning methods, such as SVM and Softmax Regression, we tried a lot over different combination of learning rate and regularization intensity. However, it had few effect on final results (only small improvement) in fact. When it comes to deep learning method, after adding the data preprocess step, the accuracy rise from 0.39 to nearly 0.8 without any other changes, which shows the model has much stronger learning ability than classical ones. What't more, there are more hyper-parameters in deep neural network model for us to conduct fine-tuning, and it is quite easy to deepen or prunning the network framework (need only a few lines of code by deep learning framework like `Pytorch` or `Tensorflow`), which show us the powerful of deep learning method in scalability and optimisation.

In general, we thought both classical machine learning methos and deep learning method have their pros and cons. We can raise strengths and avoid weaknesses by choosing proper mothods under different situations. Using the deep learning, our models can capture some essence or really important tiny features of dataset, achieve both good generalization ability and performance; while using the classical machine learning methods (logistic regression, SVM, Softmax, K-NN and so on), we can bypass the insufficiency of dataset and focus on the problem itself, find the most direct and effective classification criterions.

5. ACKNOWLEDGMENTS

Sincerely thanks to Prof. Bo Yuan and TAs for their patient guiding and assistant during our learning process. This work is successfully produced and we did benefit a lot from this course project.

6. ADDITIONAL AUTHORS

7. REFERENCES

[1] Y. Bengio, Y. LeCun, and G. Hinton. Deep learning. *nature*, 521(7553):436, 2015.

[2] H. J. Cordell. Detecting gene-gene interactions that underlie human diseases. *Nature Reviews Genetics*, 10(6):392, 2009.

[3] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, 1995.

[4] Nello Cristianini and John Shawe-Taylor. An introduction to support vector machines: and other kernel-based learning methods. 1999.

[5] M. B. Gerstein, C. Bruce, J. S. Rozowsky, D. Zheng, J. Du, J. O. Korbel, O. Emanuelsson, Z. D. Zhang, S. Weissman, and M. Snyder. What is a gene, post-encode? history and updated definition. *Genome Research*, 17(6):669–681, 2007.

[6] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *In Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS10). Society for Artificial Intelligence and Statistics*, 2010.

[7] Steven Gold and Anand Rangarajan. Softmax to softassign: Neural network algorithms for combinatorial optimization. *Journal of Artificial Neural Networks*, 1996.

[8] A. Karpathy. Cs231n: Convolutional neural networks for visual recognition. 2014.

[9] D. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint*, 1412(6980), 2014.

[10] M. Lukk, M. Kapushesky, J. Nikkilä, H. Parkinson, A. Goncalves, W. Huber, E. Ukkonen, and A. Brazma. A global map of human gene expression. *Nature biotechnology*, 28(4):322–324, 2010.

[11] H. Pearson. Genetics: what is a gene? *Nature*, 441(7092):398, 2006.

[12] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of machine learning research*, 15(1):1929–1958, 2014.

[13] A. Rechtsteiner M. E. Wall and L. M. Rocha. Singular value decomposition and principal component analysis. *In a pratical approach to microarray data analysis*, pages 91–109, 2003.

[14] Joseph Wang. Survey and summary from dna biosensors to gene chips. *Nucleic Acids Research*, 28(16):3011–3016, 2000.