Report for Bioinformatics Project, 2019 Spring

赵怿龙, 118033910139
ylzhao1996@qq.com
School of Electronic Information and Electrical Engineering, Shanghai Jiao Tong University, Shanghai, China
Shanghai, Shanghai

ABSTRACT

Data used for biology research is pretty large. It is a problem to efficiently process bioinformation. In this project, I apply two possible method to classify spices with their genes. I first used PCA to reduce dimintion and classify with SVM. Then I used neural network with four layers to do the same work. At last, I disscuessed my result of the tow method.

All my codes can be checked in https://github.com/xiaoke0515/Bioinformatic_class_project.git.

KEYWORDS

SVM, PCA, Neural Networks, Gene, Bioinformation

ACM Reference Format:
赵怿龙, 118033910139. 2019. Report for Bioinformatics Project, 2019 Spring. In *None*. ACM, New York, NY, USA, 5 pages. https://doi.org/000/000

1 INTRODUCTION

In the project, we are required to classify the samples with their genes. Main data is in the *microarray.original.txt* file. There 6000 samples and 22283 genes in the file. In *E-TABM-185.sdrf.txt*, the information of each sample are given, in which some information is used as the classification label. The project requires us to classify the samples with the genes. Two methods are required. Frist, the traditional dimensionality reduction method, such as Principal Component Analysis (PCA), Support Vector Machine (SVM) of sklearn module in python to classify. Then, the deep learning method with tensorflow is used to do the same mission. At last, the result of the two methods are compared and discussed.

None, June 16, 2019, Shanghai

ACM ISBN 000...$0.00
https://doi.org/000/000

2 METHOD

Reading File with Python

To classify the samples with the gene in file *microarray. original.txt*, we should read the data in it. It is a 5896×22283 matrix, which is too large to read into the memory of my computer. So I chose to train the classifier with batch mathod. The batch size is set by me a hundrud. Each time, I only read data of a hundrud samples in to the memory to train the classifier.

Read the Data and Category

The file *microarray.original.txt* is saved like csv style. Each row of the matrix is seperated with Return and each column is seperated by Tab. To read the file, I used the csv module of Python standard libaray. Each column of the file saves all the genes of one sample. To read a batch, the file is read from begin to end and extract the according colum. Data of the same row (the same gene) is normalized with L1 normalization:

$$x_{i,j} = \frac{x_{i,j} - \min\limits_{p=i} x_{p,k}}{\max\limits_{p=i} x_{p,k} - \min\limits_{p=i} x_{p,k}} \tag{1}$$

The file *E-TABM-185.sdrf.txt* contains the information of each sample. I am not fimiliar to the biological nouns, so I tried my best to divide the samples into some categories. The sample is divided according to the *Characteristics [DiseaseState]* column, and is divided into categories by keywords as shown in Table 1. Only 2855 samples are used to do the project, and I really do not have enough time to process the other samples. It can be a furture work.

With the 2855 labeled samples, I first divided it in to training set and testing set. 70% of the samples are chosen randomly as training set, and the other samples is used as testing set. Then each of the two sets are shuffled, so that data in the same batch will not be the same catagory.

Classify with PCA + SVM

The first mission required is to use the classical method, like PCA, logical regression and SVM, to classify the samples. I use PCA to reduce the deminsion and use SVM to classify.

PCA is a traditional method to reduce deminsion. The basis of each deminsion is need to represented as a vector. If the inner product of two vector is large, the two basis is much alike. So step of PCA is as follow. First, the data matrix

Table 1: Catagories of the Samples

Class Name	Keywords	Sample Number
Normal	normal, health	122
Leukemia	leukemia, AML	755
Atopic	atopic	64
B-cell	B-cell	221
Breast Disease	breast	856
Brain Disease	brain, Huntington	417
Bone Disease	bone, chond	150
Lung Disease	lung	77
Bladder Disease	bladder	40
Intestinal Disease	colo	153
Totally		2855

Table 2: PCA Parameters

Parameter	Value
Batch Size	100
Component	50

is normalized by its row. Then covreance matrix is achieved by:

$$\text{cov} = \frac{1}{m} X X^T \tag{2}$$

where X is the data matrix, m is the column number of X. To get the primary demension, the eigenvector of the coverianceance matrix is produced. If the eigenvalue of a eigenvector is large, the eigenvector is not much alike other eigenvector. So the eigenvectors with large eigenvalue is extraced as primary deminsion

As the data is too large, I used the *IncrementalPCA* of *sklearn* [1] module, which can realize approximate PCA with batch method. The training parameter is shown in Table 2. Each time, I feed a batch with size of 100 to fit the IncrementalPCA, with the *IncrementalPCA .fit()* method. The PCA step uses all the 5896 samples instead of the labeled 2855 sample, so that the correlation between genes can be better found. After all the data have been fed to the module once, the training of IncrementalPCA is done.

The well trained IncrementalPCA module is used to transform the data into lower diminsion. The *IncrementalPCA .transform()* method is used to perform the step. After lower the diminsion into 50, the 50×5896 matrix is saved for further classification.

Then I use the data with fewer deminsion to classify with SVM. SVM is actually a logic regression. To classify two catagories, a support vector is used, as shown in Fig. 1. The

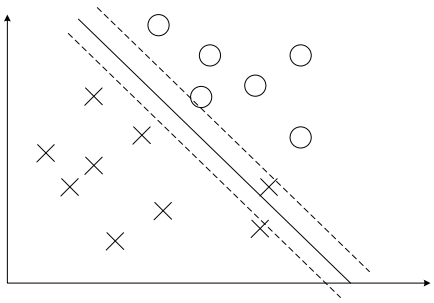

Figure 1: Principle of SVM

classifier is actually a logic function:

$$y = \frac{1}{1 + e^{-x}} \tag{3}$$

To classify two catagories, the vector should be trained (the solid line of Fig. 1) to divide the two catagories, and the margin (the dotted lines of Fig. 1) should be large enough.

I used the build-in *SVM* module of *sklearn*. The training step is realized by *svm.SVC.fit*. After PCA, the data matrix reduced to a size of 50×5896, which can be read in to the memory in one time. Therefore, no batch method is needed to fit the model.

Classify with Deep Learning Method

The other required mission is to do the same classification with deeplearning method. The Neuron Network (NN) is actually a Multilayer Perceptron (MLP) with four layers (three hidden layers and one output layer). The structure of each layer is shown in Fig. 2. Each neuron of layer i is a linear combinations of the layer $i-1$'s activations. Then activative function is applied to the linear combination. For layer i, the transmission function is:

$$a_{i,j} = \text{act}\left(\sum_{j=0}^{n} W_{i,j} \cdot a_{i-1,j} + b_i\right) \tag{4}$$

$a_{i,j}$ is the jth neuron of layer i, $W_{i,j}$ and b_i is the parameter to be trained. act is active function, where I used ReLU function:

$$\text{act}(x) = \max(x, 0) \tag{5}$$

In the output layer, the neuron number is equal to the number of catagory, which is 10 (Table 1). To get the loss of the output, softmax function is applied first, which is:

$$\sigma(x)_j = \frac{e^{x_j}}{\sum_{i=0}^{n} e^{x_i}} \tag{6}$$

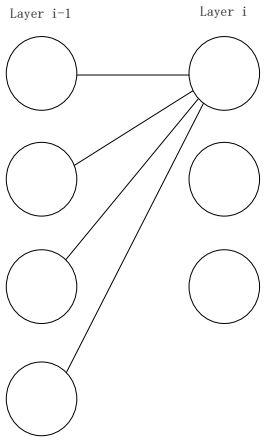

Figure 2: Structure of NN layer

then cross entrophy is applied to the output of softmax function, which is:

$$C = -[y \ln a + (1-y) \ln (1-a)] \tag{7}$$

where y is the label of dataset, a is output of softmax function. The softmax function and cross entrophy function is realized through tensorflow build-in function *tf.nn.softmax_ cross_ entropy_ with_ logits()* [2]. The reason of utlizing build-in function instead of realize it with the equation is that the build-in function can avoid the gradient ovreflow brought by the large number of weights (22283×200 in the first input layer).

To train a good classifier with the method, I tried different learning rate, different initialization configuration and different regulization decay. Then I found the best parametes which is shown in Table 3. Adam Optimizer is utilized. To converge faster, the initial learning rate is set as $1e-3$. L2 regularization is applied for the first hidden layer and L1 regularizaion is applied for other layers. Decay of regulazation is set as 1, which can better classify and maintain the balance between the training set and test set. After totally 500 iterations, a trained NN is achieved.

Because the number of sample I used is much smaller than the sample in the file. Moreover, in order to read each batch, the file is read from the first line to the last line, which cost much time. It is fatal because totally 1000 times the file is read during training. So before the training, the used batches is extracted and saved in *train_data.txt* and *test_data.txt* files. In these files, instead of representing each sample with a column, each sample is represented by a row with more than 20000 columns. Therefore, to read a batch, a continuous 100 line reading is enough. The reading time is much decreased.

Table 3: NN Parameters, also the best configuration I found after many attemps

Parameter	Value
Layer	4
Neuron Number	200, 50, 20
Active Function	ReLU
Loss Function	Cross Entrophy
Optimizer	Adam
Batch Size	100
Initial Learning Rate	$1e-3$
Regularization	L2 for layer1; L1 others
Regularization Decay	1
Iteration	500

3 RESULT

Result of PCA + SVM

Result of PCA. After PCA, the deminsion of data is lowered to 50. Eigenvalue of the 50 factors is shown in Fig. 3. Most of the eigenvalues is smaller than 5, Only about 20 eigenvalues are larger than 5.

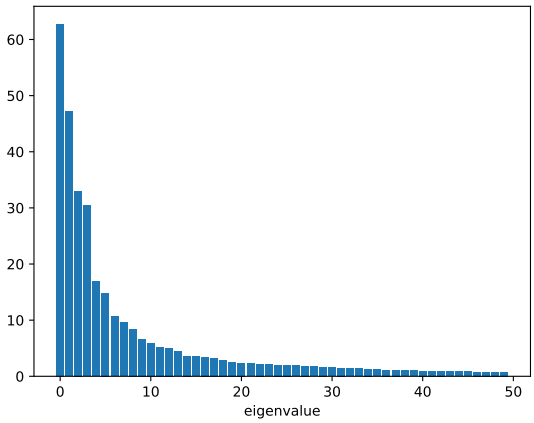

Figure 3: Eigenvalue of the 50 diminsion

Result of SVM Classification. When all the 50 eigenvalues are used, the overall accuracy and accuracy of each catagories is shown in Table 4. Both the training set and the test set have achieved a high accuracy. The accuracy of training set (97.3%) is a little higher than the testing set (94.6%). This mean that the classification is a little overfitting. Looking at each individual catagory, the largest catagories, Leukemia (550 samples for training, 205 samples for test), Breast Disease (596 samples for training and 260 for test) and Brain Disease (288 samples for training and 129 samples for test),

Table 4: Accuracy of PCA + SVM, deminsion = 50

Catagory	Train		Test	
	Number	Accuracy	Number	Accuracy
Normal	91	81.3%	31	45.2%
Leukemia	550	99.2%	205	95.1%
Atopic	43	95.3%	21	85.7%
B-cell	155	99.3%	66	94.0%
Breast Disease	596	99.5%	260	97.7%
Brain Disease	288	99.3%	129	96.1%
Bone Disease	108	96.3%	42	85.7%
Lung Disease	50	96.0%	27	81.5%
Bladder Disease	30	96.7%	10	60.0%
Intestinal Disease	117	99.1%	36	94.4%
Totally	2028	97.3%	827	94.6%

their train accuracy are over 99% , with test accuracy no lower than 95%. For small catagories, such as Bladder Disease (30 samples for training and 10 samples for test) and Normal (91 samples for training and 31 samples for test), their test accuracy are low (no more than 70%), even if the training accuracy of Bladder Disease is higher than 90%.

Varing the diminsion of training data, the accuracy of the SVM classifier is shown in Fig. 4. As we can see in Fig. 3, only 20 of the eigenvalues are larger than 5. So when varing the number of diminsion from 50 to 20, the accuracy of training set does not fall, and is still 97.3%. But meanwhile, the accuracy of test set fall from 94.6% to 89.9%. This means that the small diminsions is still important for the generalization of classifier. When deminsion reduces to 10, which means that some large deminsion is omitted, the accuracy of training set start to fall to 96.5%.

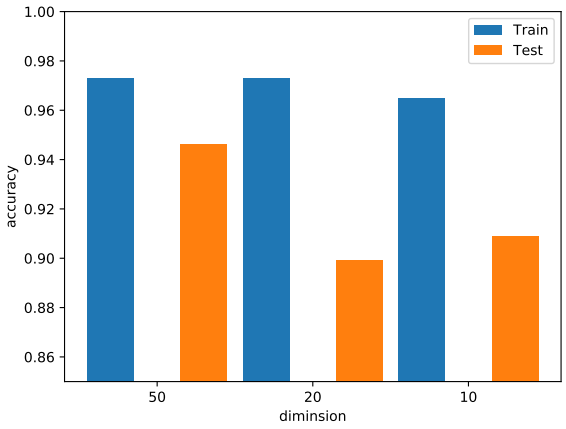

Figure 4: Accuracy varing the diminsion

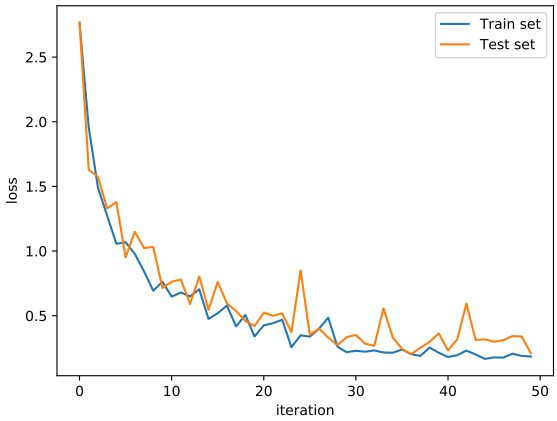

Figure 5: Loss of trainning set and test set changing during training

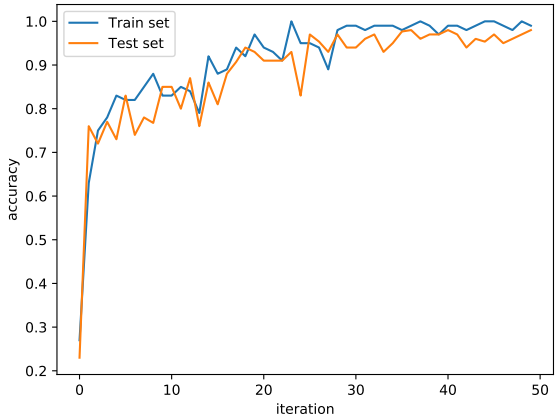

Figure 6: Accuracy of trainning set and test set changing during training

Result of Deep Learning

With the parameters in Table 3, I trained the neuron network classifier. During the 100 iterations, the loss decreased and the accuracy increased. The change of loss and accuracy is shown in Fig. 5 and Fig. 6. Both the accuracy of training set and test set increases during the training and stable at about 70%.

The result of each categories is shown in Table 5. It can be seen that for small categories, such as Bladder Disease (30 samples for training and 10 samples for test), both the training accuracy and test accuracy is larger than 95.0%. And for large categories, such as B-cell (155 samples for training and

Table 5: Accuracy of NN, number of neuron (200, 50, 20)

Catagory	Train		Test	
	Number	Accuracy	Number	Accuracy
Normal	91	96.3%	31	66.7%
Leukemia	550	99.8%	205	98.6%
Atopic	43	100.0%	21	100.0%
B-cell	155	99.4%	66	100.0%
Breast Disease	596	99.7%	260	99.5%
Brain Disease	288	100.0%	129	99.3%
Bone Disease	108	100.0%	42	100.0%
Lung Disease	50	100.0%	27	93.1%
Bladder Disease	30	96.6%	10	100.0%
Intestinal Disease	117	100.0%	36	100.0%
Totally	2028	100.0%	827	99.0%

Table 6: Result of NN Varing the number of neuron

#Neuron	Training Accuracy	Test Accuracy
(200, 50, 20)	100.0%	99.0%
(100, 50, 20)	97.0%	95.0%
(50, 30, 20)	99.0%	97.0%
(30, 20, 15)	99.0%	95%

66 samples for test), Breast Disease (596 samples for training and 260 samples for test), both the training set accuracy and test set accuracy is high, which is almost 100%. All the accuracies are higher than using traditional SVM and PCA method.

To research how the number of neuron influence the result, I varied the number of neuron of the first hidden layer. I varied the number by and the result is shown in Table 6. It can be seen that with fewer neurons, the accuracy of the network has a little decrease. But the decrease is not much. In Table 7, I present the accuracy of each categories with fewer neurons. We can find that difference between the accuracy of each categories increased.

4 DISCUESSION

From the result, both the traditional PCA + SVM method and deep learning method can classify the samples well. But for small categories, SVM + PCA is not better than deep learning method. Neuron network performs better for classifying small categories. With fewer parameters, both the two method performs worse. Even if the total accuracy does not fall too much for neuron network, the accuracy of small categories still fall evidently.

Table 7: Accuracy of NN, number of neuron (50, 30, 20)

Catagory	Train		Test	
	Number	Accuracy	Number	Accuracy
Normal	91	78.8%	31	57.1%
Leukemia	550	99.6%	205	98.6%
Atopic	43	97.7%	21	100.0%
B-cell	155	98.8%	66	100.0%
Breast Disease	596	99.8%	260	100.0%
Brain Disease	288	100.0%	129	99.3%
Bone Disease	108	93.3%	42	97.8%
Lung Disease	50	100.0%	27	82.8%
Bladder Disease	30	72.4%	10	63.6%
Intestinal Disease	117	100.0%	36	100.0%
Totally	2028	99.0%	827	97.0%

ACKNOWLEDGMENTS

Thanks for instructor of this class Prof. Bo Yuan and assistants of the class. They gave me the chance to research the project.

REFERENCES

[1] [n.d.]. Documentation of scikit-learn 0.21.2. https://scikit-learn.org/stable/documentation.html. Accessed June 8, 2019.
[2] [n.d.]. TensorFlow Document. https://www.tensorflow.org/api_docs. Accessed June 8, 2019.

.