VE492 PROJECT REPORT

BY DONG JING 515370910182

May 5, 2018

ABSTRACT

Nowadays, machine learning becomes more and more important in many fields not only in computer science due to it can actually solve some complex problems in the research of other fields.
In biology, many people do research on cells. They want to know what determines a feature of a cell. One way to know that is to use computer to learn from a group of data. Thus, machine learning provides an efficient way.
In this project, I will use classical approach PCA and SVM and deep learning to learn from a large group of data and build a model that is able to solve a binary classification problem. And I will also compare the accuracy when I set some parameters different.

1 INTRODUCTION

In the data set, there are 5896 cells and every cell has genes which are observed. In the file called 'E-TABM-185.sdrf.txt', we can see the features of the cell.
First, I use principal component analysis (PCA) to decrease the dimensions of the data. Then I choose a low dimension with most features to do support vector machine to achieve the binary classification of some features we want. Besides, I also use deep learning method to achieve the same goal.
There are many types of labels for a cell and I choose to study material types and genders.
Two common methods SVM and deep learning will be used in this project and they will be discussed later. Also, as a common method to reduce the dimension, PCA will have a big influence in this project. It does some pretreatment on the data and helps a lot in the future work.

2 METHOD

After downloading the source file, I find there are two important files called 'microarray.original.txt' and 'E-TABM-185.sdrf.txt'. One provides 22283 features for 5896 cells. The other one provides different types of label, which I want to use some methods to classify.

2.1 Principal Component Analysis(PCA)

Principal component analysis is a method to convert a set of observations into a set of values. For a set of n observations with p variables, pca can find principal component and reduce the value of n. I use matlab to achieve my goal. Matlab provides a function called *princomp* which is able to do matrix calculation of pca. This function will give several outputs. A matrix SCORE and another matrix LATENT are the things I need. In SCORE, there are still 22283 columns but they are rearranged by the importance. LATENT helps me to evaluate the importance of some features. LATENT will help me to decide reduce how many dimensions while SCORE provides the train set and test set for future work.

2.2 Support Vector Machine(SVM)

Support vector machine is a supervised learning model with associated learning algorithms. It can analyze data to classify them. The result of pca part will decide the dimension of the data that is used to analyzed in this part. Before training my SVM, I need to decide the number of samples and add label to these samples. In matlab there are two functions that will be used when we use SVM to solve a binary classification problem. One is *svmtrain* with the train set and the label as inputs. The other one is $svmclassify$ with the trained SVM and the test set as inputs. Then compare the result of $svmclassify$ with the original labels of the test set to know whether my SVM works well.

2.3 Deep Learning

Deep learning is a machine learning method that is based on learning data representations as opposed to task-specific algorithms. It can be supervised or semi-supervised or unsupervised. This time, I choose to use frame keras and deep neural network (DNN) algorithm.
Keras is a simple frame which is always used for green hand to learn deep learning. It hides the most of calculations and represents itself as a black box. Deep neural network is an artificial neural network with multiple hidden layers. It is able to solve complex problem and find non-linear relationships.
This time, I choose to use python to program. First, import data and use L2 to normalize the data. Because the organism_part and cell_line samples are unequal, I need to add weight to different samples. Then I can build the model and set some parameters. After training the data, I also test the model.

3 RESULT

3.1 PCA & SVM

3.1.1 Result of Material Type

In 'E-TABM-185.sdrf.txt', I find every cell has a feature called material type and fortunately there are only two different material type of cells, organism_part and cell_line. Thus I choose material type as the first feature that I want to use SVM to achieve the binary classification problem.

First we use *princomp* function to decrease the dimensions of the data. Then we get two important matrix SCORE and LATENT. SCORE is a 5896×22283 matrix which is the data for cells after calculation and rearrangement while LATENT is a 1×5896 matrix which shows the importance of every new dimension. After PCA dimension reduction, I get the following results: the first two columns represents 23.73% of features; 50% of features are from columns 1-14; 85% of features are from columns 1-611; 95% of features are from columns 1-2134; 99% of features are from columns 1-3986.

There is a graph shows the distributions of cells when I use PCA to decrease the dimension to 2. In the figure, the red $\star$ is on behalf of a cell with organism_part feature while the black + is on behalf of a cell with cell_line feature.

Another figure shows the distributions of cells when the dimension is 3. Similarly, the red $\star$ represents a cell with organism_part feature while the black + represents a cell with cell_line feature.

Thus, I eventually decide to use features (after PCA) 1-4000 to train the SVM structure. Also, I use 1-2 columns, 1-800 and 1-2500 columns to compare the results.
After choosing a suitable dimension which is able to represent the original data correctly and simplify the calculation, we use *svmtrain* function. Because all 5896 samples have material type and all material type of all samples are either organism_part or cell_line. So I just set the label of cell with organism_part as 1 and the label of cell with cell_line as -1. I choose to use 4000 samples as train set and the other 1896 samples as test set. Besides, for the SVM that are trained by features 1-4000, I also compare the result with the result that are from different SVM with different number of elements in train set.
There is a table showing the results.

3.1.2 Result of Gender

What is more, I also choose another type, gender. In this part, not all cells have gender and some cells have some special gender. Thus, I first find all cells that are either male or female. The label of a male cell is 1 while the label of a female cell is -1. There are 1536 cells. The dimensions of this part is also 4000. I use 1000 samples to

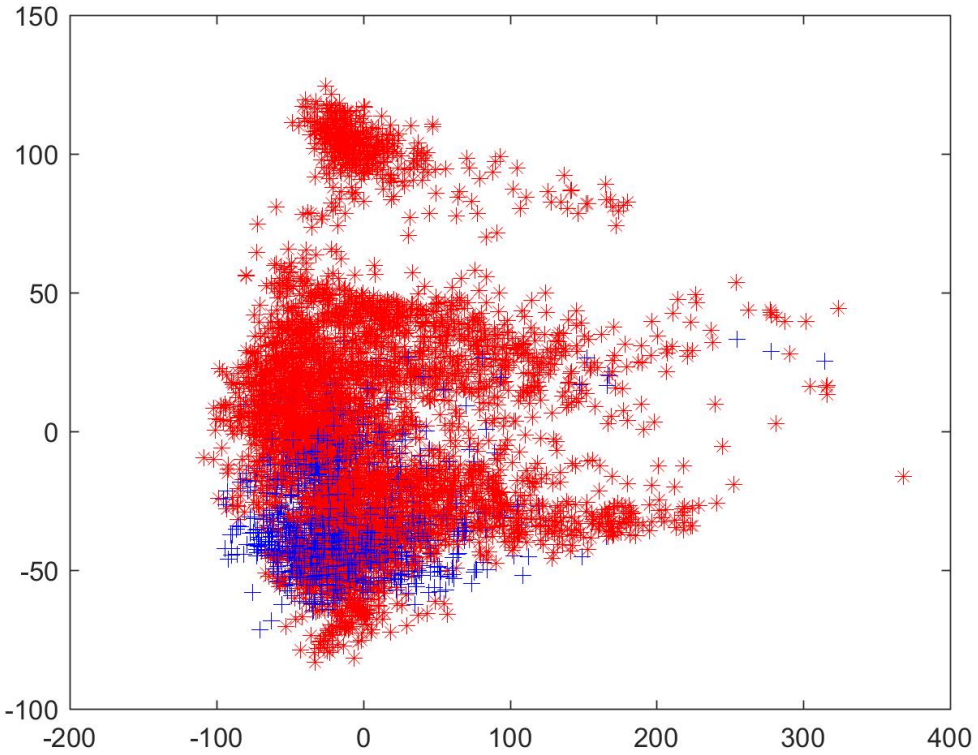

Figure 1: Distribution in 2 Dimensions.

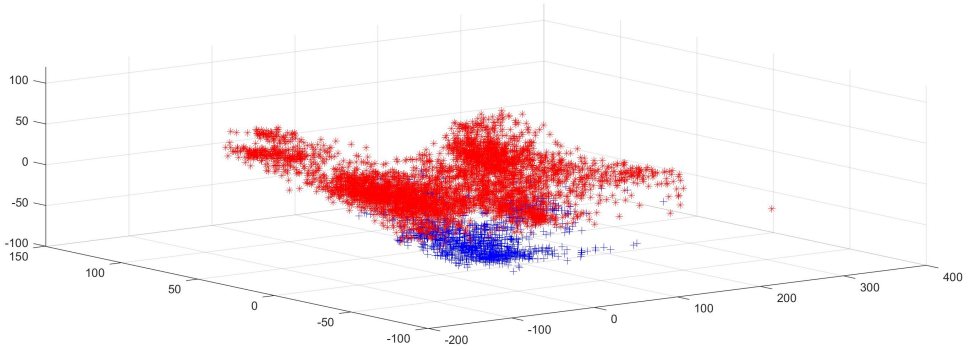

Figure 2: Distribution in 3 Dimensions.

Features	Train Set	Test Set	Accuracy
2	4000	1896	73.31%
800	4000	1896	76.97%
2500	4000	1896	79.45%
4000	4000	1896	85.76%
4000	3000	2896	82.25%
4000	5000	896	83.04%

Table 1: Accuracy of Different SVM.

train the model and use the other 596 samples as the test set. The accuracy is 75.9%.

3.2 Deep Learning

3.2.1 Result of Material Type

As mentioned before, I use frame keras and Deep Neural Network (DNN) algorithm to train a model and test the data. When I set L1 rate as zero, the accuracy of my model is about 89.67%. And if I do not balance the weight of active and positive samples, the result is changed. The first time I run the code, the accuracy of my model is 94.92% while the second time the accuracy changes to 95.04%.

3.2.2 Result of Gender

Then I import the data for gender and do the same thing again. The accuracy is 91.22% when the L1 rate is set to 0. Without balance the weight of active and positive samples, I get an accuracy of 91.44% and 92.16% (also run twice).

4 DISCUSSION

For PCA and SVM part, I find the choose of dimension is very important. A suitable number of dimensions will provide a relative accurate result in a relative short time. Besides, the choose of train set and test set is also significant. The result of gender is worse and I think the main reason is that the train set is still too small. Besides, I also find if the train set is too large so that the test set will be too small, the accuracy will also be low because one mistake will cause a big influence.

Compare the result of PCA and SVM with the result of deep learning, I find deep learning will give a model with higher accuracy than SVM. The reason may be that deep learning is able to find the inner link in the data and sometimes SVM has overfitting problem.

Totally, my result is acceptable. It shows that machine learning is able to learn from a group of data and give a model which can help scientists distinguish some types of cells. It shows the correctness of machine learning and I think if we can improve the accuracy it may be used to judge whether a person has a certain disease in the future.

APPENDIX

PCA & SVM Code

```
A=importdata('microarray.original.txt');
B=A.data';
x=zscore(B);
[~, SCORE, LATENT]=princomp(x);
s=0;
i=1;
while s/sum(LATENT)¡0.99
s=s+LATENT(i);
i=i+1;
end
label=importdata('label.xlsx');
svm_struct=svmtrain(SCORE(1:4000,1:4000),label(1:4000));
result=svmclassify(svm_struct,SCORE(4001:5896,1:4000));
num_correct=0;
i=1;
while i¡=1896
if label(i+4000)==result(i) num_correct=num_correct+1;
end
i=i+1;
end
accuracy=num_correct/1896;
```

Deep Learning Code

```
import keras
from sklearn import preprocessing
import numpy as np
import xlrd

def readData(file):
        bk=xlrd.open_workbook(file)
        sh=bk.sheet_by_name("Sheet1")
        num_row=sh.nrows
        num_col=sh.ncols
        row_list=[]
        for i in range(1,num_row):
                row_data=sh.row_values(i)
                row_list.append(np.array(row_data))
        return np.array(row_list)

traindata=readData("data.xlsx")
label=readData("label.xlsx")
```

```
testdata=readData("data2.xlsx")
label2=readData("label2.xlsx")

model=keras.models.Sequential()
model.add(keras.layers.Dense(input_dim=4000,output_dim=10000,activation='relu',
kernel_regularizer=keras.regularizers.l1(0)))
net=[10000,8000]
for len in net[1:]:
    model.add(keras.layers.Dense(output_dim=out_len,activation='relu',
kernel_regularizer=keras.regularizers.l1(0)))
model.add(keras.layers.Dense(output_dim=2,activation='sigmoid',
kernel_regularizer=keras.regularizers.l1(0)))
model.compile(optimizer='rmsprop',loss='binary_crossentropy',metrics=['accuracy'])

epochs=10
accuracy=0
for i in range(epochs):
    model.fit(traindata,label,batch_size=32,epochs=1)
    result=model.evaluate(testdata,label2)
    if result[0]¿accuracy:
        accuracy=result[0]
print(accuracy)
```