Machine Learning: Computer Exercise 2

机 53　陈声健　2015010509

1. Task Description

With the data provided last time, the new task is to do experiment on the classification of E3 and E5 data with MLP and SVM (linear kernel and Gaussian kernel). For MLP, it is encouraged to implement the algorithms by ourselves. For SVM, discussion on the effect of kernels and kernel parameters is required. An optional task is to visualize the SVs with PCA or t-SNE.

2. Experiment Design

I use tensorflow built-in function to classify E3 and E5 data with MLP. Cross validation is used to evaluate the result of the classification. The problem of overfitting is revealed.

I use the built-in SVM function of sklearn to classify the data with SVM with different kernels. And I use PCA to visualize the SVs. Experiments are carried out to explore the effect of some critical hyperparameters.

3. Method

All the experiments are run in virtualenv with python3.6. Required libraries include *tensorflow, sklearn, seaborn, pandas, numpy and matplotlib*.

In *Tensorflow*, the forward calculation the MPL is easy to implement and the backward propagation is calculated automatically. I split the dataset into training set and test set with the ratio 4:1. Different network structures are tried and the one in the code is [10, 64, 128, 64, 32, 2]. The input features size is 10 (10 genes) and the number of output classes is 2 (2 classes). The MLP is trained for 500 epochs (long enough) and the accuracy of the prediction on the test set is calculated every 10 epochs. The performance of the MLP is demonstrated in the form of learning curve and the final prediction accuracy on the test set. The code of this part is in *hw2_tf_mlp.py*.

Sklearn library provides well implemented SVM method with linear, rbf, sigmoid and polynomial kernel. Some hyperparameters like C and gamma can be customized. *Sklearn* also provides method like *classification_report* and *confusion_matrix* to evaluate the model. Additionally, PCA is also available in sklearn which can be used to visualize the SVs and *seaborn.lmplot* can be used to plot the principal components. To show why those support vectors are chosen, I visualize both the training data and the support vectors with PCA.

4. Result and Observation

The training process is run long enough (500 epoch) to demonstrate some commonly seen problems in MLP. From the plot result (Fig. 1), we can see that the training error drop down quickly in less than 100 epochs and keep at a very low value at the rest of the training time. Looking the prediction accuracy on the test set. The accuracy increases at the beginning to nearly 99% but later drops down and keep at the level of around 95%. This shows the problem of overfitting. In conclusion, if we stop the training at a suitable time, we can get nearly 100% accuracy on training set and around 99% accuracy on test set on the task of classification on E3 and E5 data. This performance is pretty good.

In SVM, I test the linear kernel with C = 0.01 and C = 100 to show the effect of C. According to the theory, C defines the penalty on misclassification. The larger C is, the more the classifier try to avoid misclassification, which will result in a harder boundary. Compare the result (Table.1) of C = 0.1 and C = 100, we can find that large C makes more accurate prediction on training set. While the result on the test set is quiet sensitive to the difference between training set and test

set. Maybe due to the quality of our dataset, I can not see smaller C consistently outperforms larger C on test set as it should be according to the theory. However, I have noticed that the accuracy can not tell the full story of C. Also, I find that large C takes much more time to fit the model to the data.

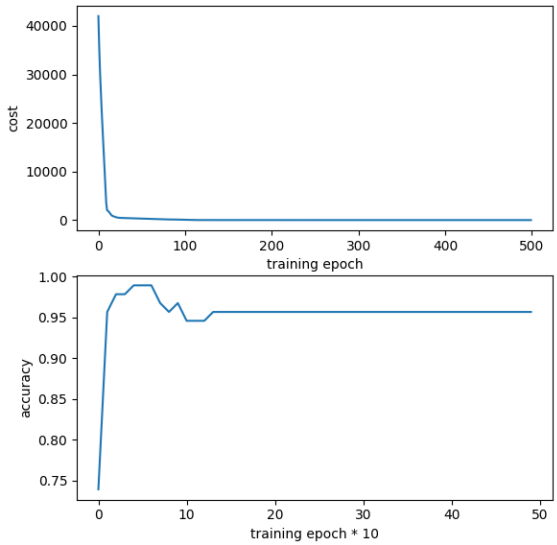

Fig. 1

label	C = 0.01	C = 100
Training Set: 0.0	0.92	0.95
Training Set: 1.0	0.97	0.97
Test Set: 0.0	0.80	0.75
Test Set: 0.1	0.94	0.95

Table.1

In rbf (Gaussian) kernel, I mainly study the effect of gamma, which defines how far the influence of a single training example reaches. Low value means far influence. Intuitively, low value can produce more general model that work better on unseen data. I carry out experiment on 3 gamma values (10e-3, 10e-6, 10e-9) and C is fixed to be 1.0. The result (Table.2) prove the intuition mentioned above. High gamma fit the training data well but perform worse on test set (misclassify all the 0-labeled test data). While low gamma solves the problem of overfitting.

label	Gamma = 10e-3	Gamma = 10e-6	Gamma = 10e-9
Training Set: 0.0	1.0	0.94	1.00
Training Set: 1.0	1.0	0.95	0.85
Test Set: 0.0	0.00	0.88	1.00
Test Set: 0.1	0.79	0.85	0.82

Table. 2

I show the result of PCA visualization in the case of linear kernel and C = 0.001 in fig. 3. I

zoom in the image to show the main idea. The full image can be found in the supplementary material. Compare the whole training set and the support vectors (part of the training set), we can observe that those support vectors roughly lay around the boundary of these two classes. That exactly corresponds to the definition of the support vector and explain why those vectors are chosen to be the support vectors. All the support vectors of each model are save in a txt file separately and attached in the supplementary materials.

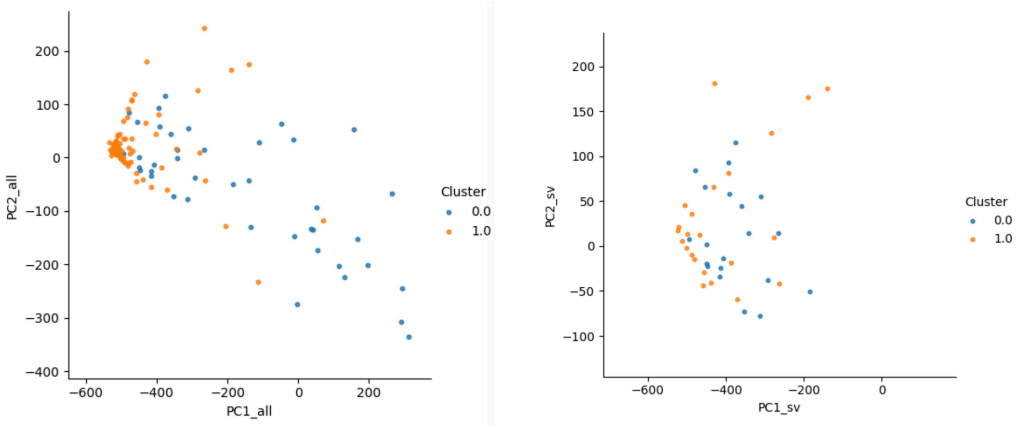

Fig.2 (left: PCA visualization of training data, right: PCA visualization of SVs)

5. Conclusion

In this exercise, I mainly explore some important characteristics of SVM: kernel, gamma, C and support vector. I try linear and rbf (Gaussian) kernels and find that rbf kernel is more powerful to classify our gene data. C parameter trade off between high classification accuracy and maximization of decision function's margin. Large C tends to have the problem of overfitting. Gamma control how far the influence of a sample can reach. Experimental result also shows that small gamma which means far reaching influence suffers less from the problem of overfitting. Visualization of the data through PCA shows that support vectors lay roughly around the boundary of two different classes.