Homework 4

Student Number: 118033910019
Name: Xingyi Wang

Problem 1. In this assignment, transfer component analysis (TCA) will be used to deal with cross-subject electroencephalography (EEG)-based emotion recognition. TCA is a typical domain adaptation method that has successfully been applied to dealing with domain shift problems.

The dataset used in this homework is the SEED dataset, which is a public EEG-based three-category emotion recognition dataset. Each subject contains 3394 samples. You need to conduct **leave-one-subject-out cross validation** to evaluate the performance of the algorithms. For each time, you need to choose one subject as the target subject (test set) and leave the other 14 subjects as the source subject (training set).

Solve the domain shift problem in the given dataset using support vector machines (SVMs). You need to finetune the parameters and only present the bset result.

Solution. In this problem, I am going to solve the domain shift problem using SVMs. The SVM models are implemented using scikit-learn Python toolbox. Each time, only one subject is left out for testing. The experimental results are shown below:

```
----------Subject 0 is left out for testing----------
Accuracy: 0.32999410724808487
----------Subject 1 is left out for testing----------
Accuracy: 0.32999410724808487
----------Subject 2 is left out for testing----------
Accuracy: 0.5810253388332351
----------Subject 3 is left out for testing----------
Accuracy: 0.6797289334119033
----------Subject 4 is left out for testing----------
Accuracy: 0.3889216263995286
----------Subject 5 is left out for testing----------
Accuracy: 0.32999410724808487
----------Subject 6 is left out for testing----------
Accuracy: 0.6487919858573954
----------Subject 7 is left out for testing----------
Accuracy: 0.6443724219210372
----------Subject 8 is left out for testing----------
Accuracy: 0.7639952857984679
----------Subject 9 is left out for testing----------
Accuracy: 0.5524454920447849
----------Subject 10 is left out for testing----------
Accuracy: 0.6832645845609899
----------Subject 11 is left out for testing----------
Accuracy: 0.6205067766647024
----------Subject 12 is left out for testing----------
Accuracy: 0.8341190335886859
----------Subject 13 is left out for testing----------
Accuracy: 0.3656452563347083
----------Subject 14 is left out for testing----------
Accuracy: 0.6378903948143784
```

Figure 1: Experimental results for SVMs

Fig. 1 shows that the accuracy of SVM in the domain shift problem can range from 32.99% to 83.41%. The mean accuracy is equal to 55.94%.

I also tried to do the feature selection using PCA in each training iteration. The results are shown in Fig. 2. We can see that PCA does not bring us good results when fewer features are selected.

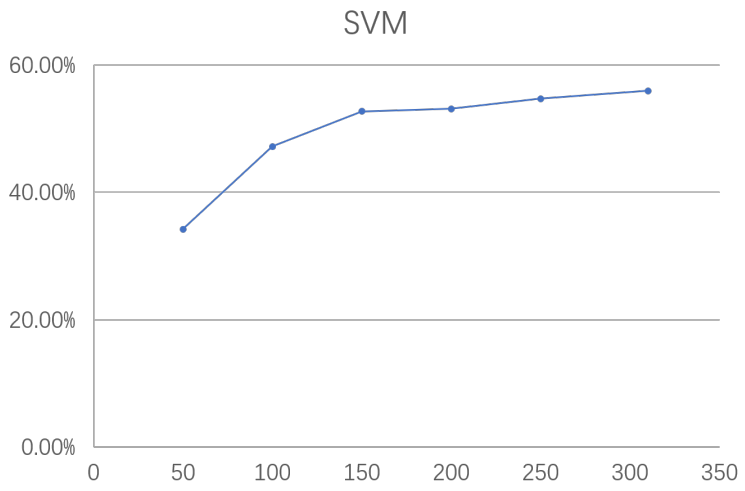

Figure 2: Comparison on different number of features based on PCA

Problem 2. Solve the domain shift problem using TCA. You need to implement TCA and use it to solve this problem. Compare the results with problem 1. Alter the latent dimension of TCA and compare the results.

Solution. In this problem, I am going to solve the domain shift problem using Transfer Component Analysis (TCA). TCA aims to map the features in both the source domain and the target domain into the same hidden domain. Then the classification can be complete in the hidden domain.

The TCA libraries for MATLAB and Python are available on Github. However, when implementing these libraries, some problems show up. In the domain mapping stage, a matrix with a size of $n \times n$ is created, where n stands for the number of samples. In the SEED dataset, the total number of samples from all domains is equal to 50910, which means that a 50910×50910 matrix is created in the TCA algorithm. This matrix is unbearably big for the common main memory. So I decide to down-sample the training data to test the TCA algorithm.

The main memory in my server can sustain at most 20000 training samples (source domain) and 3394 testing samples (target domain). However, the running time is another obstacle. I tried to run the TCA algorithm for 24 hours and the experiment for the first group (group 0 left out for testing) cannot be complete. So I am not able to estimate the computation time for all groups. I check the code for TCA algorithm, and find that the most time-consuming part is to solve the eigenvalue of the extremely big square matrix.

Finally, I down-sample the training set to 2800 samples (200 samples from each source domain), and the testing set to 2000 samples, which finally returns results to me. The results are shown in Fig. 3 and Fig. 4. As the figures show, we can get a mean accuracy of 53.40% for TCA with latent dimension 50, and a mean accuracy of 51.25% for that with latent dimension 100. These results are relatively lower than that of SVM, which may be due to

the small size of the training set. Furthermore, we can see that higher latent dimension does not provide us better results. The reason behind this may be the noise induced by the extra dimensions.

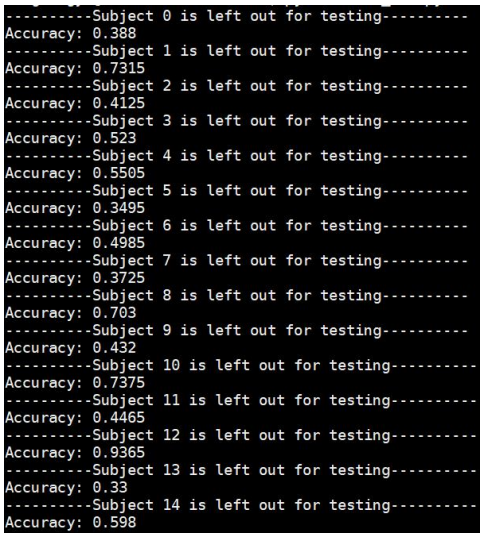

Figure 3: Experimental results for TCA with latent dimension 50

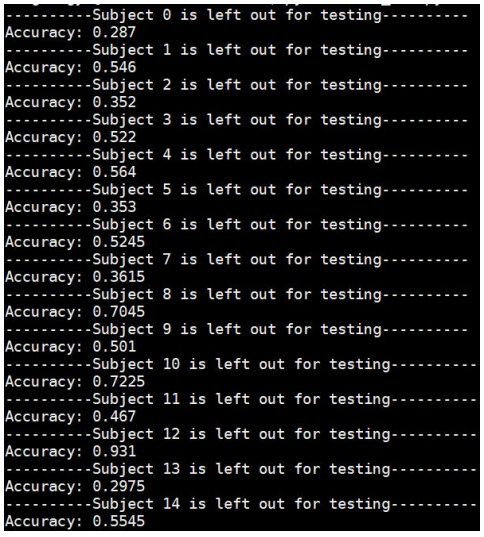

Figure 4: Experimental results for TCA with latent dimension 100

My codes in both MATLAB and Python are included in my homework zip file. The results are based on the Python version.