Project 2: Brain Computer Interfaces

Haoyun Chen
hc321@duke.edu

Abstract

Brain computer interfaces can convert human brain's signal to numerical numbers. Different pattern of the signal numbers of brains usually indicates different thought of human. This project takes various samples of human brains' signal and its corresponding thought of movement direction to train models. The models can be used to predict the humans' thought of movement from their brain signal data. Support vector machine was applied.

1. Overview

Support vector machine is a popular method in classification, which provides a robust boundary which divides the two categories to the largest theoretical distance. We applied this method in the project to classify two kinds of brain movement of human. We also discussed our results and proposed some improvements.

2. Mathematical Formulation

The decision function of support vector machine is

$$f(X) = W^T X + C \quad \begin{cases} \geq 0 & (Class\ A) \\ < 0 & (Class\ B) \end{cases}$$

Features

where W and C are parameters and X is the matrix of features in each sample. Our objective is to minimize the convex cost function with several constraints shown below.

$$\begin{aligned} \min_{W,C,\xi} \quad & \sum \xi_i + \lambda \cdot W^T W \\ \text{S.T.} \quad & y_i \cdot (W^T X_i + C) \geq 1 - \xi_i \\ & \xi_i \geq 0 \\ & (i = 1,2,\cdots,N) \end{aligned}$$

In the equations, ξ is the vector which indicates error of each sample to deal with noise. It must be larger than or equal to 0. Lambda is the regularization parameter which can be determined by cross validation. These inequality constraints and the optimization can be formulized as another optimization problem without constraints by logarithmic barriers, which is

$$\min_{W,C,\xi} \quad \sum \xi_i + \lambda \cdot W^T W - \frac{1}{t}\sum_{i=1}^{N} \log\left(W^T X_i \cdot y_i + C \cdot y_i + \xi_i - 1\right) - \frac{1}{t}\sum_{i=1}^{N} \log(\xi_i)$$
$$t \to \infty$$

To decrease round-off error, we set t to a relatively small value at first. We use Newton method to find the optimal solution of W, C and ξ in each iteration. Then we increase t by timing a constant β and enter the next iteration. The iteration stops when t is larger than a tolerance. The initial value of each iteration except the first is the optimal solution of the last iteration. In order to meet the constraints, the initial value of the first iteration can be chosen by the method that set the W(0) and C(0) to arbitrary values and set ξ(0) to:

$$\xi_i^{(0)} = \max\left\{1 - y_i \cdot \left(W^{(0)T} X_i + C^{(0)}\right), 0\right\} + 0.001$$

Let Z = [W C ξ], and the cost function f(W, C, ξ) be f(Z). The problem becomes min{f(Z)}. We use Newton method to solve this. Each iteration of the Newton method is done with following steps:

1. Compute the Newton step and decrement:

$$\Delta Z = -\nabla^2 f(Z)^{-1} \cdot \nabla f(Z) \quad \sigma = \nabla f(Z)^T \cdot \nabla^2 f(Z)^{-1} \cdot \nabla f(Z)$$

2. Stopping criterion: quit if $\sigma/2 \leq \varepsilon$ where $\varepsilon = 0.000001$.
3. Line search: choose step size s by backtracking line search. Starting from s=1, Z=Z+s*ΔZ if Z meets all the inequality constraints. Otherwise, s = s/2 and try updating again. The loop stops until the updated Z meets the constraints.
4. Update: Z=Z+s*ΔZ.

We tested the result by applying six-fold cross validation. Each time we use 5/6 of the sample set to train and test with the rest samples. Then we decide the accuracy Ac(i) each time, which equals to the number of correct predictions divided by the number of samples in the test set. Because there are six folds, we have mean and standard deviation defined as:

$$\overline{Ac} = \sum_{i=1}^{6} Ac(i)/6 \qquad stdAc = \sqrt{\frac{1}{5}\cdot\sum_{i=1}^{6}\left[Ac(i) - \overline{Ac}\right]^2}$$

In each training set, we need to have another layer of cross validation to determine the best lambda value of the current set. We use five-fold cross validation to calculate the average accuracy by taking each lambda and determine the best lambda with the highest average accuracy.

3. Experimental Results

With the methods formulated in the last section, we are able to train a support vector machine to classify the direction of brain signals. Here we show the values of W and C for the first training fold from 'feaSubEOvert.mat'. For *W*, we only show the five most dominant values with the largest magnitude.

W(1)	W(2)	W(3)	W(4)	W(5)	C
0.0014	0.0013	0.0011	0.0010	0.0009	-1.1324

Table 1: W and C values of a training fold.

The channel weights plot of W in the first training fold is shown below.

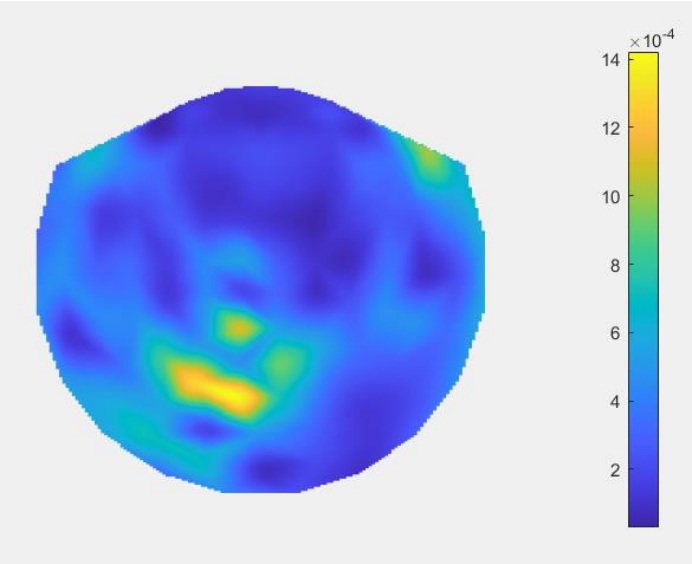


Figure 1: channel weights plot of a trained W

The area with lighter color shows that the W is relatively higher, which means this area is more important for prediction. In other words, this area of X will have larger impact on deciding whether the brain's signal pattern is left or right.

Then we show the test accuracy of each fold in a table.

Ac(1)	Ac(2)	Ac(3)	Ac(4)	Ac(5)	Ac(6)	Mean	STD
0.9	1	1	1	0.95	0.925	0.9625	0.044

Table 2: the accuracy of training with 'feaSubEOvert.mat'.

Ac(1)	Ac(2)	Ac(3)	Ac(4)	Ac(5)	Ac(6)	Mean	STD
0.85	0.8	0.8	0.975	0.875	0.925	0.8708	0.0697

Table 3: the accuracy of training with 'feaSubEImg.mat'.

4. Discussion

From the two above tables we found that the average accuracy of 'feaSubEImg.mat' is much lower than that of 'feaSubEOvert.mat'. This means the characteristics from the second dataset are relatively 'vague' than that from the first one and can be harder to predict, which also indicates characteristics of the 2 classes from 'feaSubEImg.mat' is relatively closer. We can verify this by doing the following experiment. First, we calculate the mean of the characteristics of the 2 classes from both datasets. Then we calculate the l2 distance of the 2 mean characteristics from each dataset. Finally, we compare the distance. The dataset with smaller distance will be harder to predict. The code is shown below:

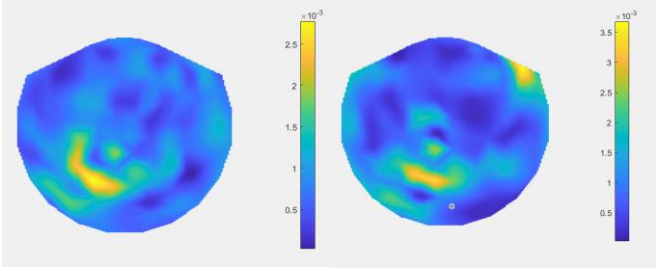
```
Img = load('feaSubEImg.mat');
Img1 = Img.class{1,1};
Img2 = Img.class{1,2};
meanI1 = mean(Img1,2);
meanI2 = mean(Img2,2);
Overt = load('feaSubEOvert.mat')
Overt1 = Overt.class{1,1};
Overt2 = Overt.class{1,2};
mean01 = mean(Overt1,2);
mean02 = mean(Overt2,2);
distImg = norm(meanI2-meanI1)
distOvert = norm(mean01-mean02)
```

The result shows that the distance of 'feaSubEImg.mat' is 1280.9, which is smaller than the distance of 'feaSubEOvert.mat' (1809.8). The standard deviations of the 2 classes of each dataset are also compared and show no large difference. As a result, this experiment can be the indicator of the difficulty of classification and can be used to explain for the reason of changes in model accuracy.

Observing that the formalized cost function f(Z) of the problem is high dimensional. So, when applying the Newton method, the computational cost could be extremely high if we let the computer calculate the numerical solution of Gradient and Hessian matrices. As a result, a tough work which is required to do is to calculate the analytical solution by hand. Then we formulate the solution in the form of matrix calculation by code, and we can get gradient and Hessian only by substituting all the parameters. This saved much computing time.

There are several factors that can affect the accuracy. First, we can inspect more lambda values to perform better regularization. In my program, instead of timing lambda by 100, I multiply lambda by 3 in each iteration, and the accuracy is improved by about 2%. Second, the choice of initial t does not have to be very small. An initial t as large as 1000 is enough to guarantee the function to be smooth and very small round-off error. If we change t from 1000 to 1, the accuracy raises smaller than 1%, and the run time is longer. Also, the parameter 'Tmax' and 'tol' are factors which impact the accuracy. Theoretically, 'Tmax' should be larger and 'tol' should be smaller to achieve higher accuracy. However, the values given by the requirements of the project are appropriate enough so that we do not need to modify them.

Another finding is that from the channel weights plot, only small area of the plot has lighter color, with higher value of W. Then we make the channel weights plot of other training sets in the same dataset. We found that the high values of W always concentrate on specific area.



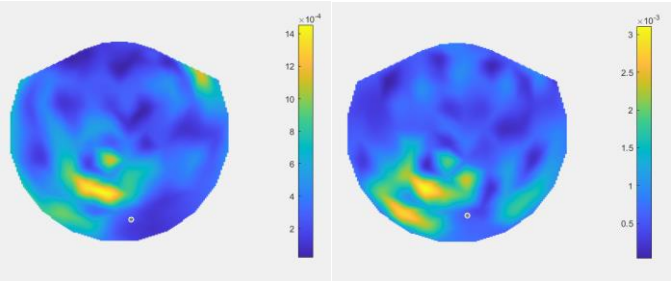


Figure2: channel weights plot of different training sets in the same dataset. The high values all concentrate on the left-bottom part.

This indicates that most X values in each sample does not affect the result of prediction. As a result, it can be feasible for us to do dimension reduction for X, e.g., the PCA algorithm, which can greatly reduce the run time without impact the accuracy so much.

References

[1] DKU ECE580 lecture notes 16&17, Xin Li.