Project 2: Brain-Computer Interfaces

Hongrui Guo
steven.guo@duke.edu

Abstract

This project uses the interior point method to train a support vector machine (SVM) to classify a dataset of electroencephalogram (EEG) signals with binary labels. The optimal weight is found by using the Newton method and line search. This report would also cover how the accuracy is improved and some other findings.

1. Overview

In this project, a MATLAB program was developed for classifying multi-channel EEG signals collected by a brain-computer interface (BCI) device. EEG signal is a type of electrical bio-signal collected from the subject's scalp for showing the activities of the surface layer of the subject's brain. A BCI is a computer system that collects brain activities, analyzes them, and outputs some commends based on them. In this experiment, BCI is used for recording the brain activities of the subject.

The dataset consists of 2 classes: left and right. 120 trials are recorded for each class (240 trials total). This makes the dataset balanced and easier to classify. Each trial contains the recordings from the sensors of the BCI. Each sensor is placed on a different location of the scalp (a 3D space) and the collected 204 features are flattened to a 1D array. The data is normalized with z-score normalization in the preprocessing phase.

The classification method used for this experiment is SVM. To train the SVM, the margin, distance between the plus and minus planes need to be maximized. This problem is formulated as a convex optimization problem. Since complex data like EEG is not linearly separable, a soft margin is used to train an SVM with noise by adding errors in the constraints. Then this problem is solved by using the interior point method. For each iteration for the interior point method, the unconstrained nonlinear optimization is solved by the Newton method with line search to adjust the step size.

Section 2 will explain how to formulate this classification problem as a convex optimization problem, how the optimal values are derived, and how they are been constraints to their domain. Section 3 will present the weights from the classifier, the performance of the classifier, and some other collected data. Section 4 will discuss the results, analysis, and improvements to the algorithm.

2. Mathematical Formulation

The initial soft margin maximization problem can be formulated as the following:

$$\min_{W,C,\xi} \sum \xi_i + \lambda W^T W$$

$$\text{S.T.} \quad y_i(W^T X_i + C) \ge 1 - \xi_i$$
$$\xi_i \ge 0$$
$$(i = 1, 2 \dots, N)$$

The inequity constraints can be merged into the objective function using the indicator function.

$$I(u) = \begin{cases} 0 & (u \le 0) \\ +\infty & (u > 0) \end{cases}$$

Since the indicator function is not smooth, it is approximated using a logarithmic barrier.

$$I(u) \approx -\frac{1}{t}\log(-u) \quad (u \le 0)$$

The initial optimization problem can then be formulated as the following unconstraint problem:

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

A feasible solution would satisfy the following conditions:

$$\begin{cases} y_i(W^{(0)T} X_i + C^{(0)} > 1 - \xi_i^{(0)}) \\ \xi_i^{(0)} > 0 \end{cases} (i = 1, 2 \dots, N)$$

The initial guess $Z^{(0)} = [W^{(0)} \ \ C^{(0)} \ \ \xi^{(0)}]$, and $[W^{(0)} \ \ C^{(0)}]$ was set to some arbitrary number and $\xi^{(0)}$ is assigned by:

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

A small number of 0.001 is added to prevent $\xi^{(0)}$ being set to the end, so ξ would not be 0.

After setting the initial value, the initial t is set to 1 and the optimal solution can be obtained. Then the initial value is reset to the optimal solution $[W^* \ \ C^* \ \ \xi^*]$, and $t = \beta t$ where $\beta = 15$. This process is repeated until $t \ge T_{\max} = 1000000$.

For obtaining the optimal solution the Newton method is used to approach the optimal solution. The Newton step and the decrement are computed as the following:

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

Z is updated by $Z = Z + s\Delta Z$ until $\frac{\sigma}{2} \le \varepsilon$ where $\varepsilon = 0.000001$.

The step size s is determined using the line search. s is initialized to 1. Then it is updated by $s = 0.5 \cdot s$, until $Z + s \cdot$

$\Delta Z \in dom(Z)$. In other words, it is repeated until the following conditions are met:

$$\begin{cases} y_i(W^{(0)T}X_i + C^{(0)} > 1 - \xi_i^{(0)}) \\ \xi_i^{(0)} > 0 \end{cases} (i = 1, 2..., N)$$

This ensures that Z would never step outside of its domain and the initial constraints are always met.

2 classes in the dataset are assigned with the label $y_i = -1\ or\ 1$. The dataset is divided into 6 parts for k-fold cross-validation to find the average accuracy.

In each fold, the training set is divided into 5 parts (in each fold, 1 part for testing, 4 parts for training) for a 2nd level cross-validation to find the optimal λ, the penalty weight. Then the optimal λ is used to construct the SVM for the current fold using 1 part for testing, and 5 parts for training.

3. Experimental Results

Results from feaSubEImg.mat:

	W1	W2	W3	W4	W5	C
value	-0.3922	-0.3185	-0.3008	0.2937	-0.2885	-0.8770
location	155	152	153	137	141	NaN

Table 1.1: 5 most dominant W with the largest magnitude and C for 1st fold

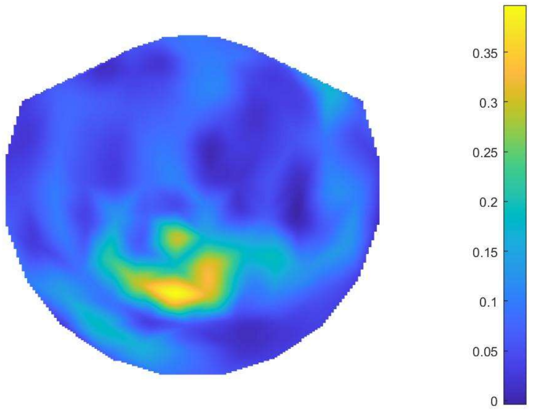

Figure 1.1: 2D plot of the magnitude of W for 1st fold

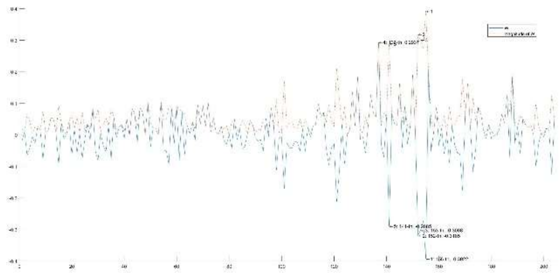

Figure 1.2: 1D plot of W and its magnitude, with five most dominant values with the largest magnitude for 1st fold

Fold	1	2	3	4	5	6	mean	std
Accuracy	0.875	0.925	0.925	0.900	0.925	0.925	0.9125	0.0209

Table 1.2: test accuracy of each fold, and their mean and standard deviation

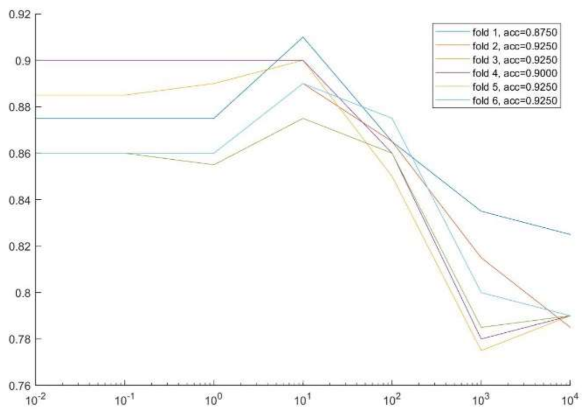

Figure 1.3: test accuracy of each fold for $\lambda = [0.01, 0.1, 1, 10, 100, 1000, 10000]$

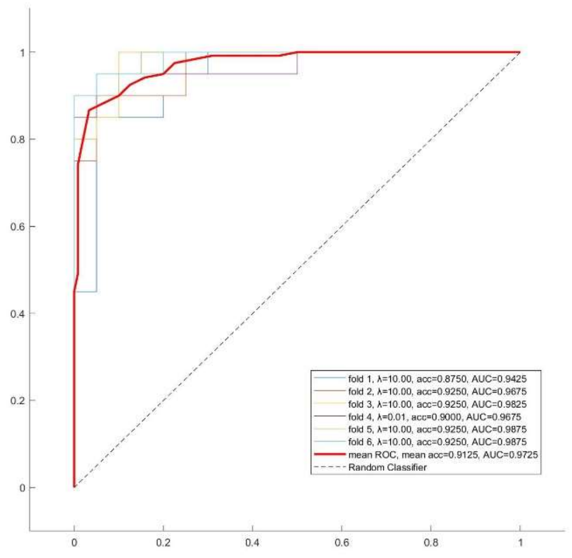

Figure 1.4: ROC curve for each fold

Results from feaSubEOvert.mat:

	W1	W2	W3	W4	W5	C
value	-0.2402	-0.2246	0.2177	-0.1983	0.1766	-0.5379
location	155	141	137	101	156	NaN

Table 2.1: 5 most dominant W with the largest magnitude and C for 1st fold

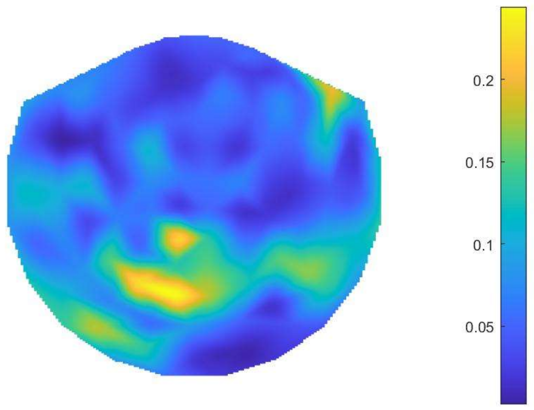

Figure 2.1: 2D plot of the magnitude of W for 1st fold

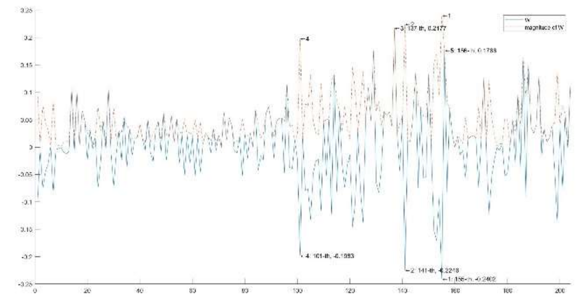

Figure 2.2: 1D plot of W and its magnitude, with the five most dominant values with the largest magnitude for 1st fold

Fold	1	2	3	4	5	6	mean	std
Accuracy	0.975	0.975	0.950	0.950	0.925	0.900	0.9485	0.0292

Table 2.2: test accuracy of each fold, and their mean and standard deviation

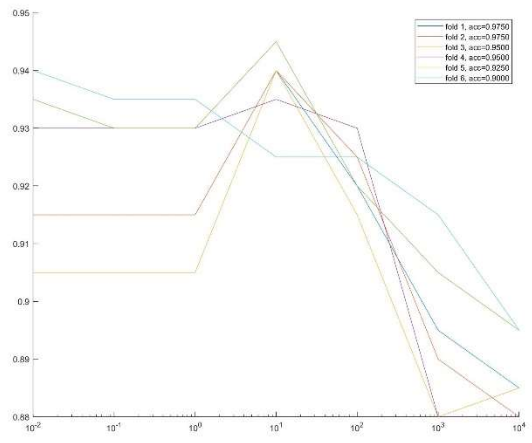

Figure 1.3: test accuracy of each fold for $\lambda = [0.01, 0.1, 1, 10, 100, 1000, 10000]$

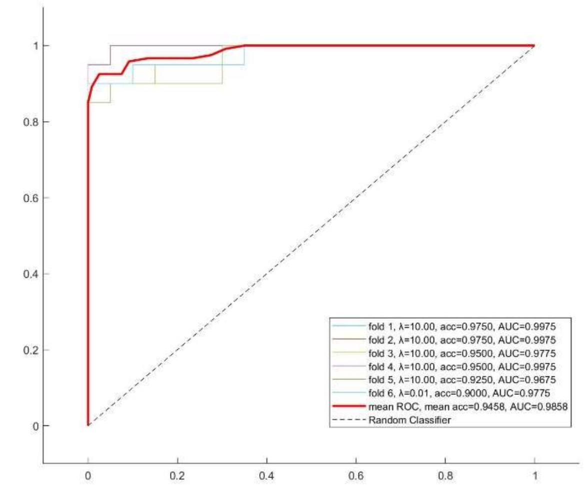

Figure 1.4: ROC curve for each fold

4. Discussion

The first thing that can be observed is that the second dataset can produce a better classification. The mean accuracy for feaSubEImg.mat is 0.9125 and the mean accuracy for feaSubEOvert.mat is 0.9485. The 2nd one's accuracy is 3.6% higher than the first one and the area under the ROC curve is also higher than the 1st one. This may be caused by subject difference, experiment setup, sensors, or other possible factors.

The second thing that can be seen is that the five most dominant values for both datasets include features with indexes 137, 141, and 155. This can mean that the brain sections under sensors that produced these 3 features are strongly related to directions and navigation. It might be possible to classify the direction with only these 3 features. To validate this guess, an array with 1 on index 137, 141, and 155 and 0 for the rest is plotted.

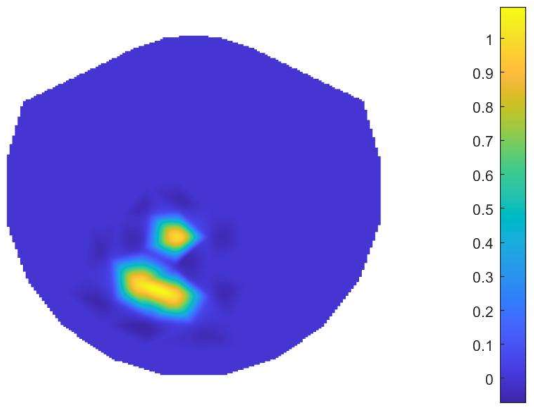

Figure 3: 2D plot of the array with 1 on 137, 141, and 155 and 0 for the rest

It is clear that the highlighted maximum area in this plot matches the area for the plots for both W. This means the other features are relatively trivial compare to them.

One limitation of the interior point approach is that the speed of this algorithm can be rather slow with multiple layers of the nested while loop. To speed up the program, the cost function doesn't use any loops for computing the function value, gradient, and the Hessian matrix. Since MATLAB can compute matrix operations very fast, the program can achieve a reasonable speed regardless of the high cost of the interior point method. Another solution for this is to solve the dual problem directly instead of using the interior point method.

From the λ vs. accuracy plot, we can see that the optimal λ is around 10 in general. The λ value can be further refined by running another experiment with $\lambda = [7, 8, 9, 10, 11, 12, 13]$.

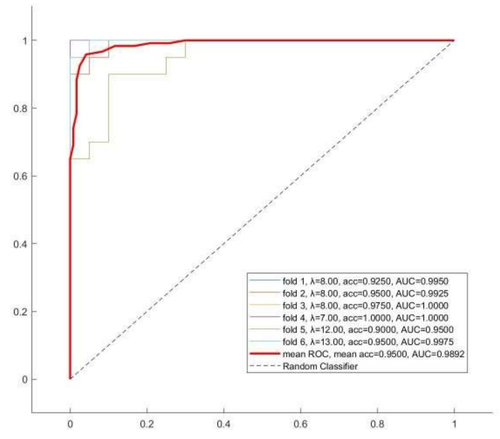

Figure 4: ROC curve for each fold, with $\lambda = [7, 8, 9, 10, 11, 12, 13]$ on dataset feaSubEOvert.mat

Then we can see that the classifier performs better around $\lambda = 7\ or\ 8$. Next, another experiment is done with $\lambda = [6.5, 6.8, 7, 7.2, 7.5, 7.8, 8, 8.2, 8.5]$.

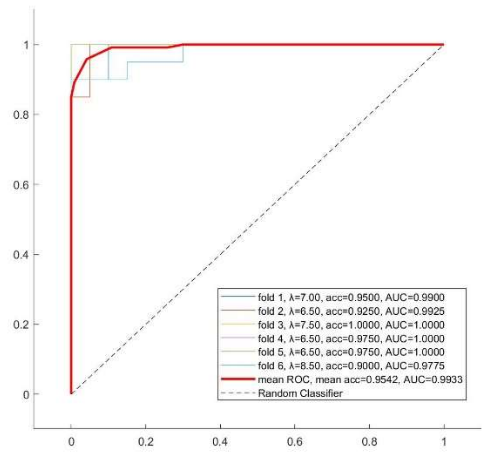

Figure 5: ROC curve for each fold, with $\lambda = [6.5, 6.8, 7, 7.2, 7.5, 7.8, 8, 8.2, 8.5]$ on dataset feaSubEOvert.mat

We can see that the optimal λ is around 6.5. This process can be repeated to approach the exact optimal solution. But for the scope of this experiment, we can say that the optimal $\lambda = 6.5$.

Another improvement made to improve the performance of the classifier is to select the optimal λ using the largest AUC of the ROC curve instead of the highest accuracy. By doing so the performance of the classifier including accuracy would be improved and the optimal λ or its range can be found more easily with the same set of λ candidates.

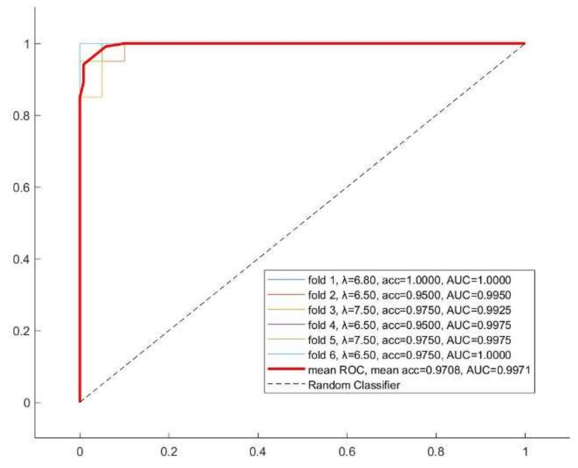

Figure 6: ROC curve for each fold, with $\lambda = [6.5, 6.8, 7, 7.2, 7.5, 7.8, 8, 8.2, 8.5]$ on dataset feaSubEOvert.mat using AUC of ROC as the metric

References

[1] ECE 580 lecture slides by Dr. Xin, Li