Project 3: Movement Decoding for Brain Computer Interfaces

<Ye Zhang>
<yezhang@andrew.cmu.edu>

Abstract
In this project, I implemented a two-class SVM classifier and applied the SVM classifier to decode BCI features into two classes (i.e. Left vs. Right)

1. Overview

I implemented SVM classifier. Firstly, I convert the maximum boundary problem into a convex optimization problem. Then I implemented interior point algorithm to solve this optimization problem. In each iteration, I solved unconstrained nonlinear problem using Newton method. I obtain the optimal λ for each training fold and test the accuracy on the testing fold. To obtain the best λ, I also use cross validation inside each training fold.

2. Mathematical Formulation

First, I construct the convex optimization problem for the maximum margin problem:

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

S.T. $y_i(W^T X_i + C) \geq 1 - \xi_i$

$\xi_i \geq 0$

$(i = 1,2...,N)$

Then I convert this problem into a logarithmic barrier problem
:

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

$t \to \infty$

To solve this problem, I used interior point algorithm:
First, I select an initial value t and an initial guess for W, C, and ξ_i for each training instance.
Then I solve the unconstrained nonlinear optimization problem using Newton Method. Then I update W, C and ξ_i
by the newly calculated result
, and also increase t by 15 times. I repeat this procedure until t is larger or equal 100000.

To solve the nonlinear optimization problem, I use Newton method. Firstly I give an initial feasible guess to satisfy the following:

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

And then I set the corresponding W and C to an arbitrary value and assign $\xi^{(0)}$ to the following:

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

Using these initial values, I compute the Newton step and the decrement as follows:

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

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

where Z = [W,C, ξ]

If $\sigma / 2 \leq 0.000001$, I quit the iteration. Otherwise, I choose an optimal step size s by backtracking line search and update Z as: $Z = Z + s\Delta Z$, and repeat this procedure until $\sigma / 2 \leq 0.000001$.

For backtracking line search, I set the initial s as 1. Then I repeat the following:
If the following is satisfied, then quit.

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

Otherwise, update s= 0.5s, $Z = Z + s\Delta Z$
The whole process is as follows:
In the first level cross validation, I choose one fold as testing data, and the other five folds as training data. To obtain the optimal λ, I use the second lever cross validation. I partition the training data into 5 folds. Then I choose one fold as the testing data, and the other 4 folds as the training data. For each optional λ, I use the second level 5 folds cross validation to calculate the accuracy. I choose the best λ to test the accuracy in the first level testing data.

3. Experimental Results

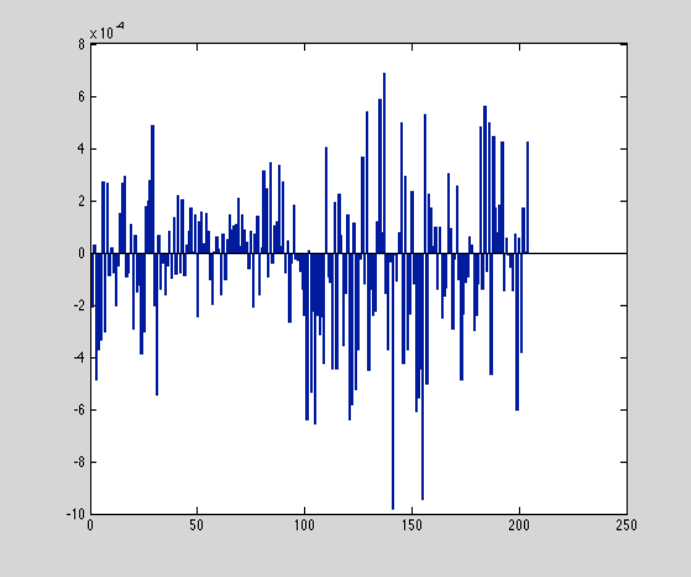
feaSubEOvert:
the test accuracy is:

1	2	3	4	5	6
0.85	1	0.975	1	0.90	0.90

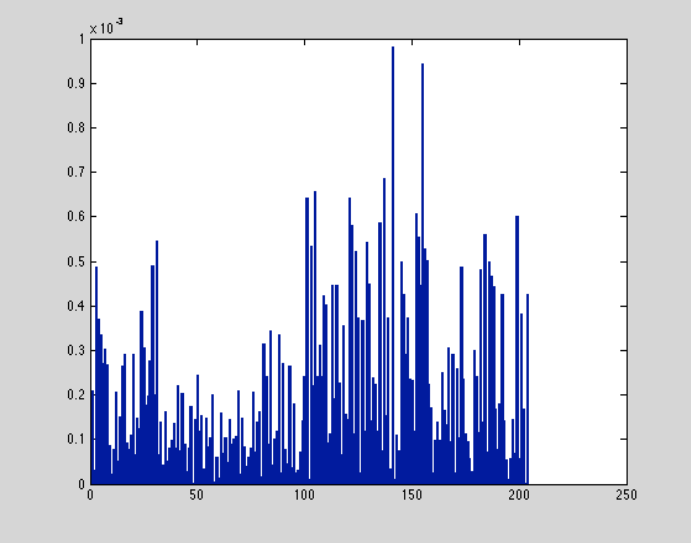
Show the mean accuracy and standard deviation of all folds.
The mean: 0.9375
The standard deviation is: 0.0627.
The top 5 |W|:

1	2	3	4	5
9.7997e-4	9.4242e-4	6.8694e-4	6.5512e-4	6.4165e-4

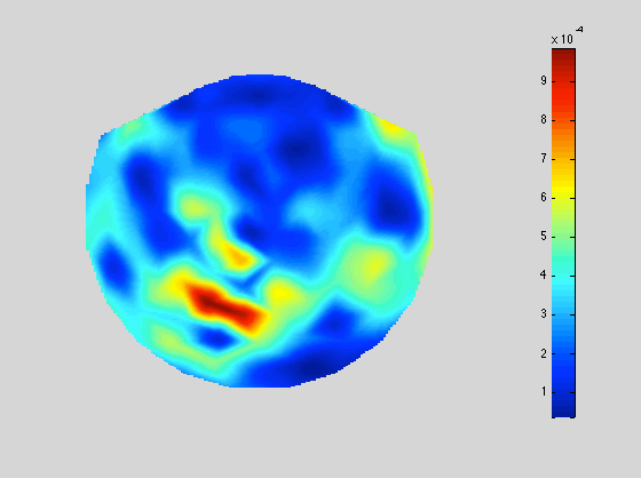
C = -0.6433
The plot for W of 204 features:



The plot for |W| of 204 features:



The channel weight plot:



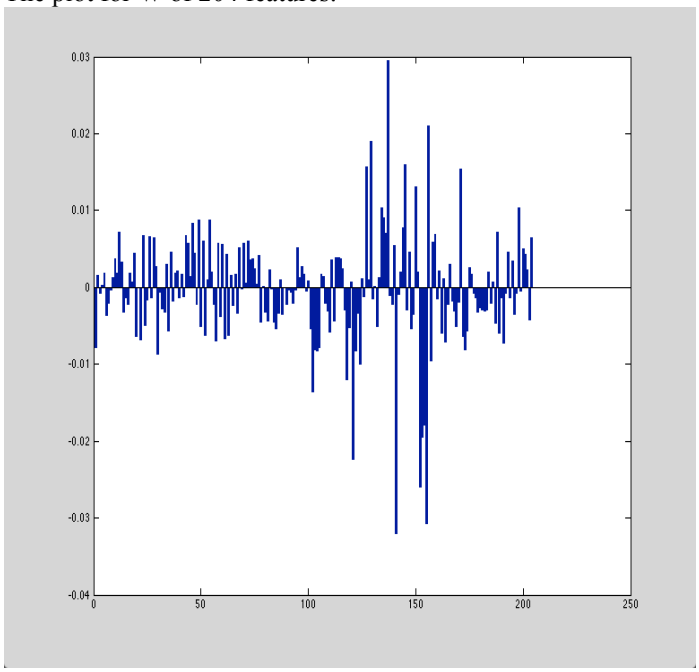
feaSubEImg:
the test accuracy for 6 training fold:

1	2	3	4	5	6
0.85	0.775	0.825	0.975	0.9	0.9

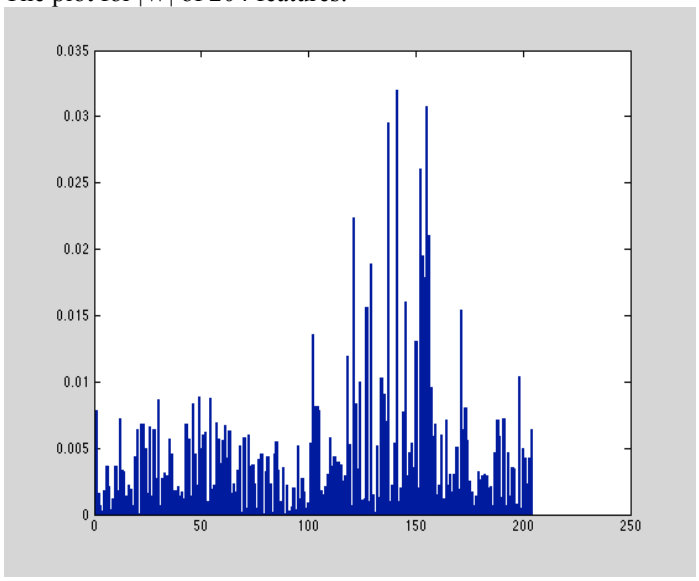
The mean accuracy: 87.08%
The standard deviation is: 0.0697.
The top 5 |W|:

1	2	3	4	5
0.0295	0.0260	0.0224	0.0210	0.0195

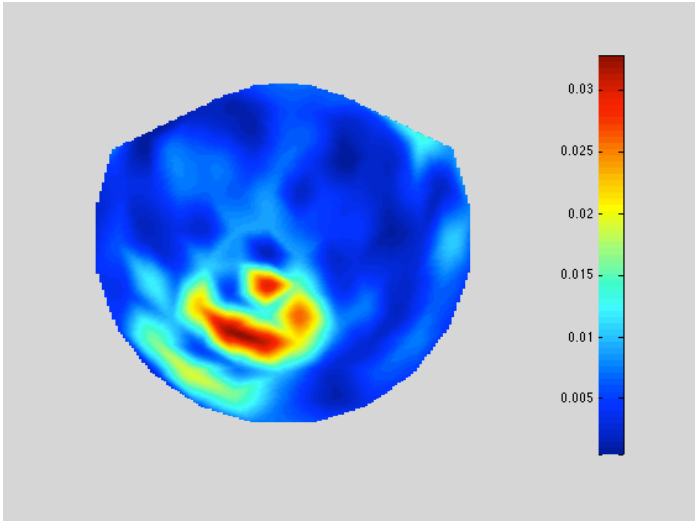
The plot for W of 204 features:



The plot for |W| of 204 features:



C = -33.76
The channel weight plot:



4. Discussion

Please try to interpret your experimental results. Topics may include, but not limited to:

- Factors that may impact classification accuracy.
  The parameter λ.
  The larger λ is, the more errors in the training data are allowed by the soft margin classifier. The smaller λ is, the fewer errors in the training data can be tolerated by the soft margin, and thus the easier over fitting will happen in the testing phase.
  Whether the feature values are normalized before the training.
- Limits or problems of your approach
  The implementation of Newton Method is not optimal.
- Possible improvements that can be done
  Calculate Hessian Matrix in a more compact way.
  Another improvement might be firstly normalizing the dataset to make different features in the same range.
  The number of possible λ
  can be increased, and not limited to the given 4 numbers.

  Anything unique you have done to improve/validate your program's accuracy/efficiency
  When implementing Newton method, I observed the change of decrement in each iteration, to make sure that it converges.
  When implementing backtracking line search, I observed the change of Z and $\xi^{(k)}$ to make sure that it converges.

  I tried different strategies to partition the whole dataset to ensure that the accuracy makes sense.

References

[1] Use an enumerated list here for any references, such as books or journal/conference papers.