Introduction to Machine Learning

Mini-Project 2

Brain-Computer Interface Movement Decoding

Libo Zhang (lz200)

Table of Contents

1. Introduction (Problem Description)

(1) What is the nature of this project?

Brain-computer interfaces (BCIs) can function as practical communication tools between the brain's electrical activity and external devices including but not limited to computers, wheelchairs, unmanned aerial vehicles, and mechanical arms [1]. By utilizing/decoding/classifying high-dimensional electroencephalogram (EEG) signals, BCIs can help patients who do not have neuromuscular control to interact with the outside world [1].

This project makes use of support vector machines (SVMs) to build linear (and nonlinear) classifiers, with the purpose of correctly classifying whether a BCI user wants to make a left movement or a right movement, given high-dimensional EEG signals with 204 features. In this project, briefly speaking, we should appropriately utilize SVMs to complete a binary classification task.

1. Introduction (Problem Description)

(2) What are the goals of this project?

There are several goals that should be achieved at completion. These goals are shown below.

Firstly, explain, analyze, and demonstrate how to mathematically formulate the BCI movement decoding task as a binary classification problem, given the high-dimensional EEG data.

Secondly, explain, analyze, and demonstrate why and how we should apply SVMs with appropriate regularization to solve the formulated binary classification problem, given the high-dimensional EEG data.

Thirdly, implement two-level cross-validation. In the second-level cross-validation, we should optimize the regularization parameter for SVMs. In the first-level cross-validation, we should use the optimal regularization parameter found in the second-level cross-validation to optimize the classification accuracy.

Fourthly, compare SVM classification performances and optimized regularization parameters between using the imagined data set and using the overt data set. Interpret experimental results including but not limited to reporting the number of support vectors (SVs) identified by SVMs, visualizing weights on the brain surface, indicating dominant weight channels, showing classification accuracies and plotting receiver operating characteristic (ROC) curves for each fold of the first-level cross-validation.

1. Introduction (Problem Description)

(3) What is the brief summary of key results?

Several key results are summarized below.

Firstly, the BCI movement decoding task can be mathematically formulated as a binary classification problem.

Secondly, we can apply SVMs with appropriate regularization to successfully solve the formulated binary classification problem, with relatively good classification accuracies for both imagined and overt data sets.

Thirdly, we can achieve better classification performance using the overt data set than the imagined data set. In specific, when visualizing feature weights on the brain surface, using the overt data can induce higher contrast between the area of dominant channels and other nondominant area.

Fourthly, for linear SVMs (no kernel), L-1 norm (Lasso) regularization can result in better total cross-validated accuracies than L-2 norm (Ridge) for both the imagined and overt data sets. However, SVMs with Lasso regularization are much more computationally expensive to achieve convergence during two-level cross-validation.

Fifthly, introducing nonlinearity to SVMs (radial basis function (RBF) kernel) does not help improve classification performance, compared with linear SVMs.

2. Mathematical Formulation

(1) How to formulate BCI movement decoding as a binary classification problem?

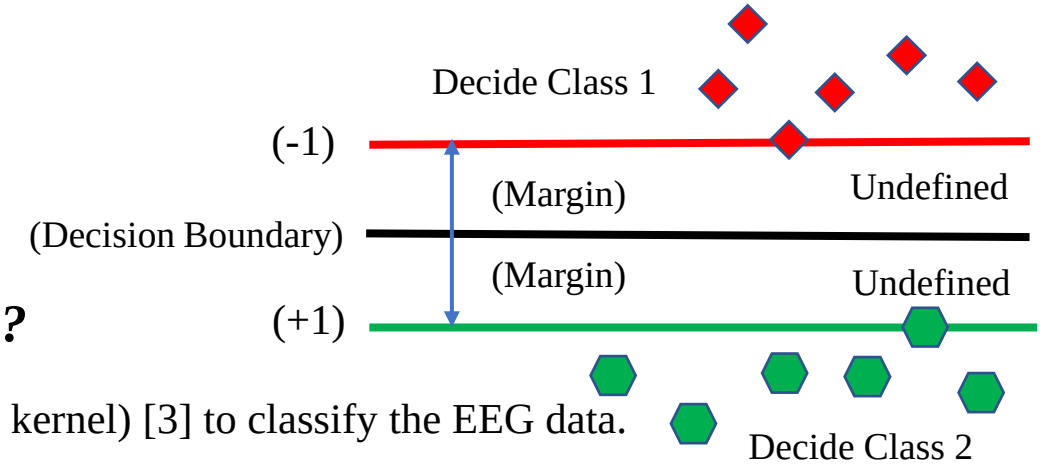
Given the imagined and overt data sets, each EEG observation (trial) has 204 electrodes, or equivalently 204 features. Therefore, we can define each EEG observation as $X = [x_1, x_2, \dots, x_{204}]^T \in \mathbb{R}^{204\times 1}$.

Utilizing 204 features, we want to train a machine learning model to classify whether one EEG observation represents a left movement (Class 1, or H_0) or a right movement (Class 2, or H_1). If we regard this machine learning model as a decision function $f(X)$, which takes each EEG observation as input and calculates a scalar decision statistic $\alpha \in \mathbb{R}$, then we can classify this observation based on a predefined threshold $\beta \in \mathbb{R}$ [2].

$$\begin{cases} f(X) = \alpha \\ Decide\ H_0\ if\ \alpha < \beta \\ Decide\ H_1\ if\ \alpha \geq \beta \end{cases}$$

Based on the above mathematical formulation, **we can successfully formulate the BCI movement decoding task as a binary classification problem.** However, what machine learning algorithms should we use to appropriately define the decision function and the threshold?

2. Mathematical Formulation



(2) How to apply a linear SVM to classify the EEG data?

As formally required by this project, we should build a linear SVM (no kernel) [3] to classify the EEG data.

(I will explain *why* an SVM is a good choice for this classification problem after this *how* section.)

Firstly, this is a binary classification problem, so we can mathematically define class 1 (H_0) as the target variable ($y = -1$) and define class 2 (H_1) as the target variable y $= +1$. For a linear SVM, we should assume the EEG data is linearly separable, so each feature should correspond to a weight parameter, and finally we should add a bias term [3]. Based on linearity, we should have 204 weights defined as $W = [w_1, w_2, \dots, w_{204}]^T \in \mathbb{R}^{204\times 1}$ and 1 bias parameter defined as $b \in \mathbb{R}$. We can also define the distance from one observation to the decision boundary as $f(X) = W^T X + b$, which should be a scalar value. Here we define threshold $\beta_1 = -1$ for class 1 and $\beta_2 = +1$ for class 2, then we can classify one observation based on the judgement shown below.

$$\begin{cases} f(X) = W^T X + b = \alpha \\ Decide\ H_0(-1)\ if\ \alpha \le -1 \\ Decide\ H_1(+1)\ if\ \alpha \ge +1 \\ (Undefined)\ if\ -1 < \alpha < +1 \end{cases}$$

Then, when training a linear SVM classifier with many observations, how should we perform optimization?

2. Mathematical Formulation

(2) How to apply a linear SVM to classify the EEG data?

To answer the proposed question in the previous slide, firstly, for all training observations (denoted as $i = 1,2,\dots,N$), we define their minimum distance to the decision boundary as margin, denoted as $M \in \mathbb{R}$, so we can have the following equation [4].

$$M = \min_i \left[\frac{y_i(W^T X + b)}{\|W\|}\right] = \frac{1}{\|W\|}\min_i[y_i(W^T X_i + b)], i \in \{1,2,\dots,N\}, X_i \in \mathbb{R}^{204\times 1}, y_i \in \{-1,+1\}, W \in \mathbb{R}^{204\times 1}, b \in \mathbb{R}$$

The reason for defining the "minimum" distance is that we want support vectors of each class to locate and determine its boundary, while trying to incorporate all other observations of the same class. Subsequently, a linear SVM classifier should be trained to maximize this margin. Here the reason for maximizing M is that we want linearly separate/classify training observations of different classes as accurately as possible [3]. In other words, we want to optimize 204 weight parameters $W \in \mathbb{R}^{204\times 1}$ and 1 bias parameter $b \in \mathbb{R}$ to achieve maximation of $M \in \mathbb{R}$, therefore our optimization goal can be defined as

$$\underset{W,b}{\operatorname{argmax}}(M) = \underset{W,b}{\operatorname{argmax}}\left\{\frac{1}{\|W\|}\min_i[y_i(W^T X_i + b)]\right\}, i \in \{1,2,\dots,N\}, X_i \in \mathbb{R}^{204\times 1}, y_i \in \{-1,+1\}, W \in \mathbb{R}^{204\times 1}, b \in \mathbb{R}$$

To completely define an optimization problem, we also need to impose some constraints which are very important if we want to practically solve this optimization problem [5]. Here we suppose that for all training observations, the minimum distance to the decision boundary is predefined to be 1. This means that all identified support vectors should have a distance equal to 1, while the other "unimportant" observations have a distance greater than. In a word, the constraint can be denoted as

$$y_i(W^T X_i + b) \geq 1, i \in \{1,2,\dots,N\}, X_i \in \mathbb{R}^{204\times 1}, y_i \in \{-1,+1\}, W \in \mathbb{R}^{204\times 1}, b \in \mathbb{R}$$

2. Mathematical Formulation

(2) How to apply a linear SVM to classify the EEG data?

Now our optimization problem can be formulated as

$$\underset{W,b}{\operatorname{argmax}}\left(\frac{1}{\|W\|}\right)\ such\ that\ y_i(W^T X_i + b) \geq 1, i \in \{1,2,\dots,N\}, X_i \in \mathbb{R}^{204\times 1}, y_i \in \{-1,+1\}$$

If we take the reciprocal values of $(\frac{1}{\|W\|})$, we can also apply L-1 norm (Lasso) regularization with parameter λ_1 to penalize large weights in terms of absolute scale and induce sparsity [6]. Then we can rewrite the formulation as

$$\underset{W,b}{\operatorname{argmin}}(\lambda_1\|W\|)\ such\ that\ y_i(W^T X_i + b) \geq 1, i \in \{1,2,\dots,N\}, X_i \in \mathbb{R}^{204\times 1}, y_i \in \{-1,+1\}$$

However, a linear SVM should solve a convex optimization problem, therefore we need to further take the square values of $(\|W\|)$, which means currently we should apply L-2 norm (Ridge) regularization as the correct regularization [7]. Suppose the L-2 norm regularization parameter is λ_2, now we can rewrite the formulation as

$$\underset{W,b}{\operatorname{argmin}}(\lambda_2\|W\|^2 = \lambda_2 W^T W)\ such\ that\ y_i(W^T X_i + b) \geq 1, i \in \{1,2,\dots,N\}, X_i \in \mathbb{R}^{204\times 1}, y_i \in \{-1,+1\}$$

Now we have successfully formulated the linear SVM convex optimization problem, which can be efficiently solved by quadratic programming [8]. Therefore, **we can conclude that we have successfully applied a linear SVM to classify the EEG data.**

2. Mathematical Formulation

(3) Why an SVM is a good choice for the BCI movement decoding binary classification problem?

Given the imagined and overt data sets, we only have 240 EEG observations for each data set, and each observation has 204 features, which means we are dealing with high-dimensional data. In addition to high dimensionality, when implementing two-level cross-validation (which I will explain in detail in the section of Experimental Results), we only have 200 training observations for each first-level cross-validation, and 160 training trials for each second-level cross-validation fold, which means we have more features for each trial than the number of training trials! Therefore, we should indeed be worried about the curse of dimensionality, and we should realize that the high dimensionality of EEG data could induce overfitting problem or result in strange/unexpected relationship between the high-dimensional features and the true classification targets/labels [9].

Recall that the decision boundary of an SVM is determined by a few support vectors, which are part of total training observations. Once these support vectors can be captured/defined, we can determine the decision boundary of an SVM and complete training. All other training observations which are not support vectors will not affect the decision boundary of an SVM, therefore even if we remove these additional data points, the SVM classification performance determined by a few support vectors will not be distorted. Therefore, given the large number of features (204) and relatively small number of training trials (200 or 160), if we want to mitigate negative effects induced by the curse of dimensionality and to achieve relatively good classification performance, we should choose an SVM to work on this BCI movement decoding binary classification problem.

In summary, we have successfully explained ***why* an SVM is a good choice for this classification problem.**

3. Experimental (Simulation) Results

Before presenting all experimental results, I want to briefly describe my experimental (simulation) conditions.

For the first-level cross-validation, I have $240 = 120 \times 2$ (120 for class 1 and the other 120 for class 2) total EEG observations given either the imagined or overt data set. I divide 240 trials into 6 folds, 5 for first-level training and 1 for first-level testing. I assign $40 = 20 \times 2$ (20 for class 1 and the other 20 for class 2) EEG trials for each fold, so I have $5 \times 20 \times 2 = 200$ training observations and $1 \times 20 \times 2 = 40$ testing trials for each iteration. I apply the optimal regularization parameter found in the second-level cross-validation to the single first-level testing fold for each iteration, to optimize the classification accuracy.

For the second-level cross-validation, I have $5 \times 20 \times 2 = 200$ total EEG observations obtained from the 5 first-level training folds. Here I divide 200 trials into 5 folds, 4 for second-level training and 1 for second-level testing, so I have $4 \times 20 \times 2 = 160$ training observations and $1 \times 20 \times 2 = 40$ testing trials for each iteration. My goal for the second-level cross-validation is to optimize the regularization parameter for L_2 (Ridge) regularization. I consider and test a large range of regularization parameters shown below.

$$\lambda = [10^{-6}, 10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 10^{0}, 10^{1}, 10^{2}, 10^{3}, 10^{4}, 10^{5}, 10^{6}]$$

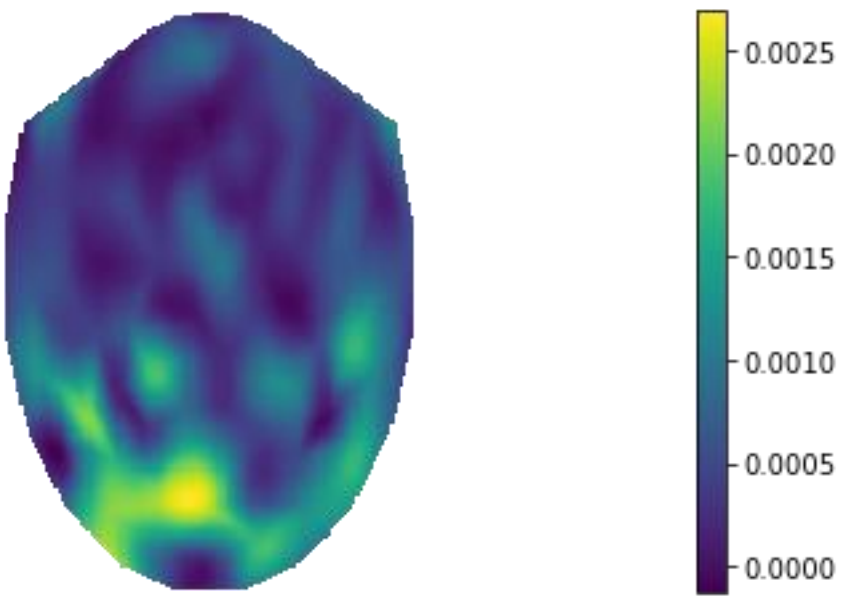
Here I want to first define some abbreviations for reading convenience. I abbreviate “receiver operating characteristic” to “ROC”. I abbreviate “probability of detection” to “PD”. I abbreviate “probability of false alarm” to “PFA”.

Next, I will present all experimental (simulation) results **for applying linear SVMs (no kernel) with L_2 (Ridge) regularization.**

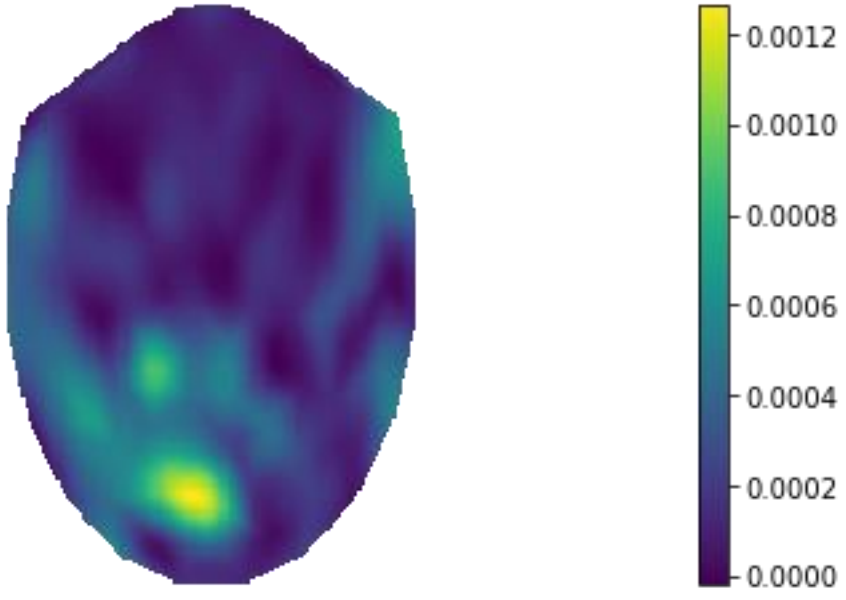
3. Experimental (Simulation) Results

a) For the 1^{st}-level cross-validation fold #1

i. Visualize the weights on the brain surface (using the provided $show\_chanWeight$ function)



Imagined Movement (Linear SVM, Ridge)



Overt Movement (Linear SVM, Ridge)

Key take-away points: The overt movement has higher contrast between the dominant area and other non-dominant area than the imagined movement. Here the dominant area is defined by weights with large absolute scale values.

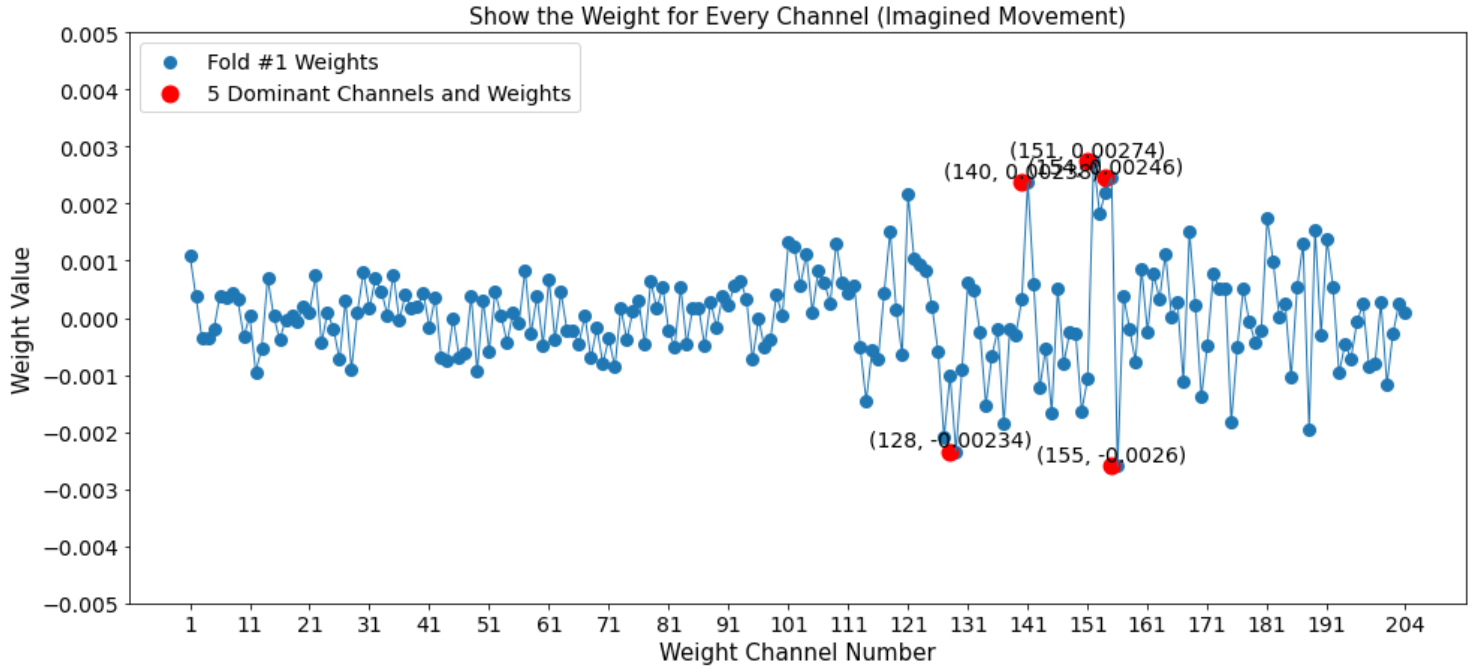
3. Experimental (Simulation) Results

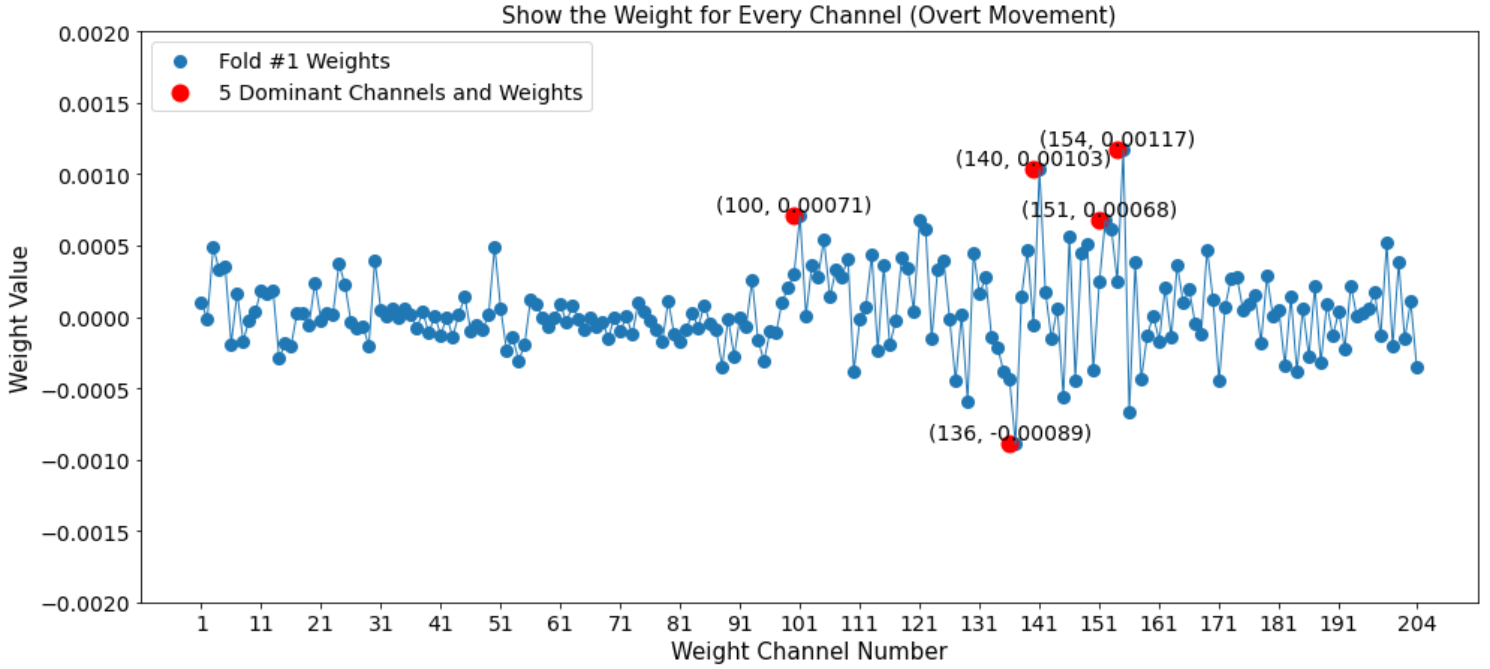
a) For the 1^{st}-level cross validation fold #1

ii. Show (plot) the weight for every channel

(A list of 5 dominant channels and their weights will be clearly reported in a table)

Key take-away points: The overt movement has a larger difference between the 5 dominant weights and other weights than the imagined movement. For the overt movement, many weights are very close to 0, but are not exactly equal to 0, so the weights are not sparse. The low sparsity is what we would expect by applying Ridge regularization.





3. Experimental (Simulation) Results

a) For the 1^{st}-level cross-validation fold #1

iii. Provide a list of the 5 dominant channels and their weights (channels are counted from 1 to 204)

Channel Number	Weight Value
152	+0.00274
156	-0.00260
155	+0.00246
141	+0.00238
129	-0.00234

Imagined Movement (Linear SVM, Ridge)

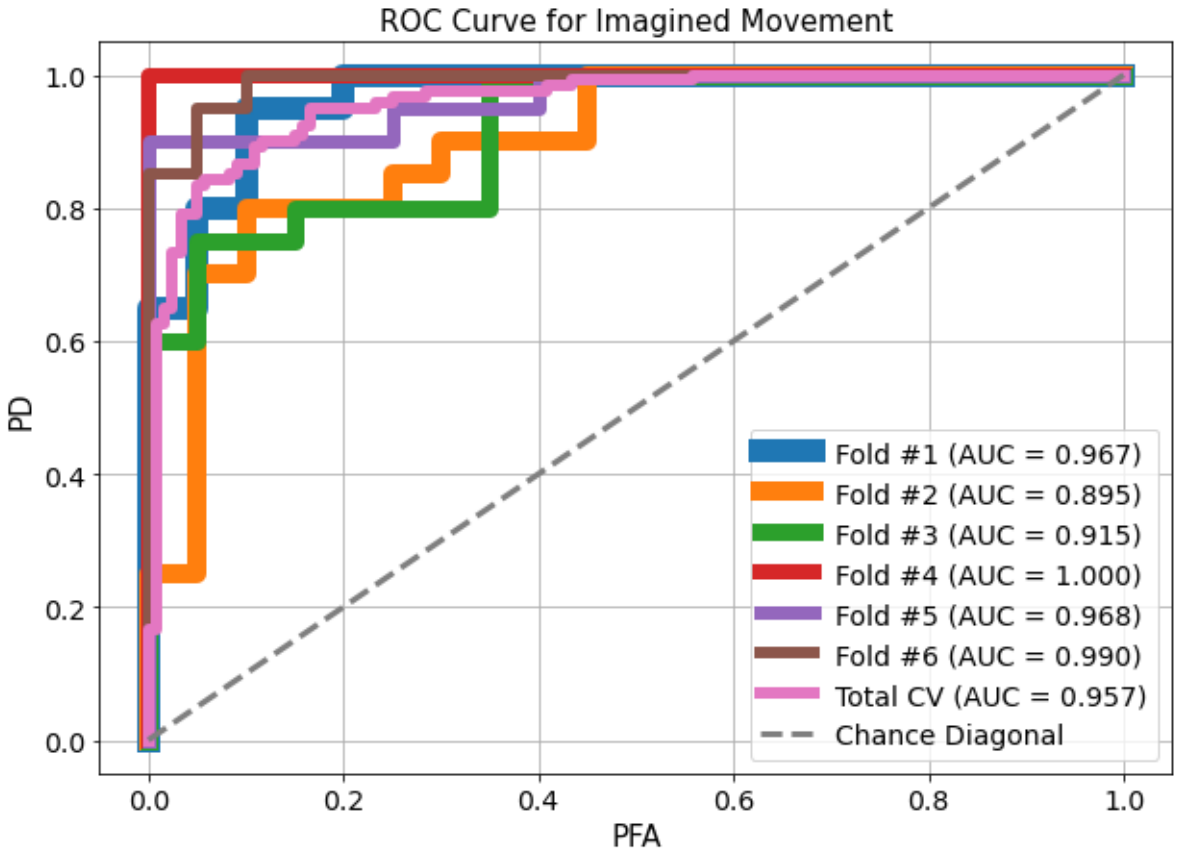
Channel Number	Weight Value
155	+0.00117
141	+0.00103
137	-0.00089
101	+0.00071
152	+0.00068

Overt Movement (Linear SVM, Ridge)

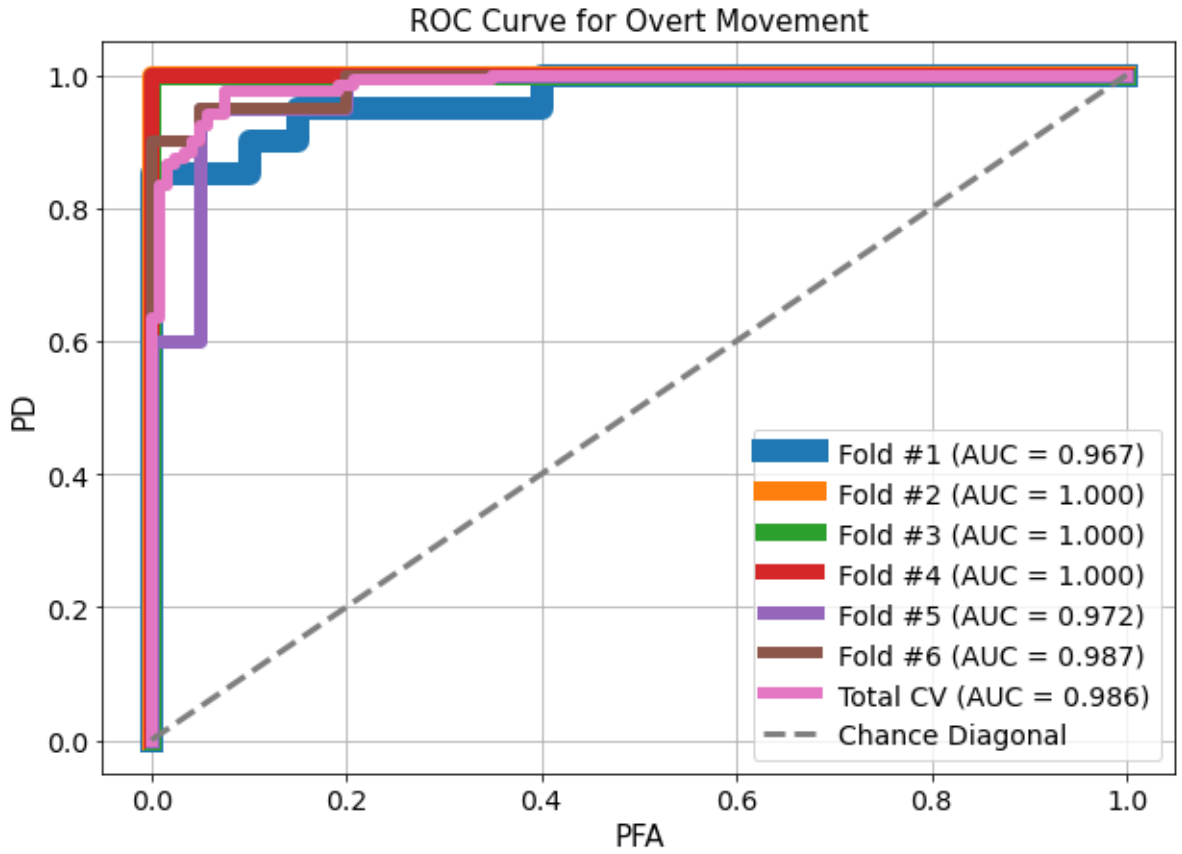
Key take-away points: Although the imagined movement has larger absolute scale of dominant weight values, we can see that channel number 141, 152, and 155 are listed for both imagined and overt movement, which means these 3 channels (features) are very important in determining whether a left movement or a right movement should be made.

3. Experimental (Simulation) Results

b) Provide the ROC for each 1^{st}-level cross-validation fold (6 ROCs), and the total cross-validated (CV) ROC on a single graph



Imagined Movement (Linear SVM, Ridge)



Overt Movement (Linear SVM, Ridge)

Key take-away points: For the imagined movement, the individual fold ROCs are not representative of the total ROC.
For the overt movement, the individual fold ROCs are representative of the total ROC. In addition, the overt movement has a larger area under curve (AUC) value for the total cross-validated ROC than the imagined movement, which means we can achieve better classification performance with the overt movement.

3. Experimental (Simulation) Results

c) Provide accuracy for each 1^{st}-level cross-validation fold assuming a threshold $\beta = 0$ (6 accuracy values), and the total cross-validated accuracy

	Accuracy
Fold #1	0.85
Fold #2	0.80
Fold #3	0.825
Fold #4	0.975
Fold #5	0.90
Fold #6	0.925
Total Cross-validated	0.8792

Imagined Movement (Linear SVM, Ridge)

	Accuracy
Fold #1	0.90
Fold #2	0.95
Fold #3	0.975
Fold #4	0.975
Fold #5	0.90
Fold #6	0.95
Total Cross-validated	0.9417

Overt Movement (Linear SVM, Ridge)

Key take-away points: For the imagined movement, the individual fold accuracies are not representative of the total accuracy. For the overt movement, the individual fold accuracies are representative of the total accuracy. In addition, the overt movement has a larger total cross-validated accuracy than the imagined movement, which means we can achieve better classification performance with the overt movement.

3. Experimental (Simulation) Results

Compare the L_2 (Ridge) regularization parameter values and the number of support vectors across the two data sets.

	Optimal Ridge Regularization Parameter	Number of Support Vectors
Fold #1	100000 = 10^5	41
Fold #2	100000 = 10^5	46
Fold #3	1000000 = 10^6	63
Fold #4	100000 = 10^5	49
Fold #5	100000 = 10^5	43
Fold #6	100000 = 10^5	49

Imagined Movement (Linear SVM, Ridge)

	Optimal Ridge Regularization Parameter	Number of Support Vectors
Fold #1	1000000 = 10^6	33
Fold #2	1000000 = 10^6	44
Fold #3	100000 = 10^5	38
Fold #4	100000 = 10^5	39
Fold #5	100000 = 10^5	34
Fold #6	100000 = 10^5	36

Overt Movement (Linear SVM, Ridge)

Key take-away points: The optimized Ridge regularization parameters for both the imagined and overt data sets are very large, and the overt movement has one more 1000000 parameter. The overt movement has relatively smaller number of support vectors for individual folds than the imagined movement.

3. Experimental (Simulation) Results

To earn a 9.5 on this section, after presenting all experimental results of applying linear SVMs with L_2 (Ridge) regularization, I also want to apply linear SVMs with L_1 (Lasso) regularization to both the imagined and overt data sets to see if I can improve the classification performance by inducing model sparsity. During my experiments, however, I encountered some convergence problems with very large and very small regularization parameters. Therefore, compared with L_2 (Ridge) regularization, here I need to narrow down the range of candidate regularization parameters. For Lasso regularization, I consider the following λ values

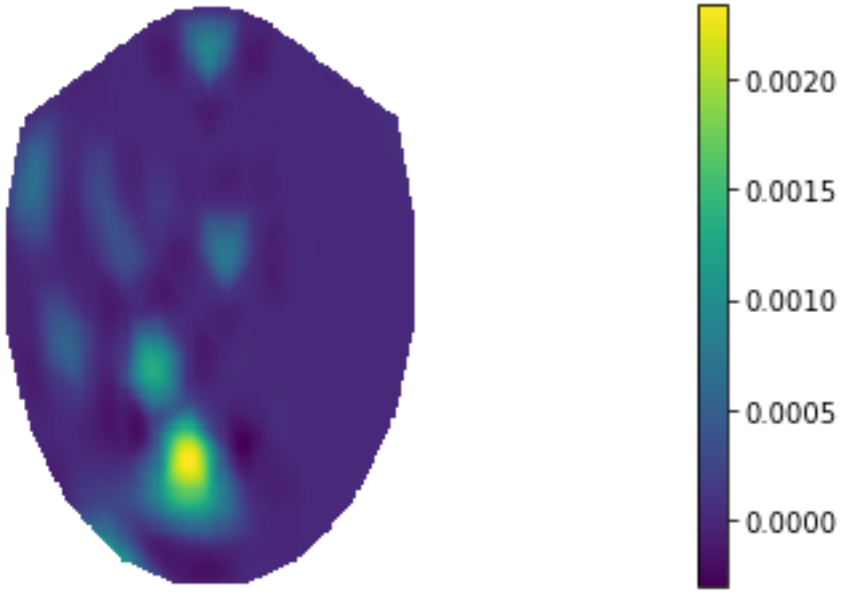
$$\lambda = [\ 10^{-3}, 10^{-2}, 10^{-1}, 10^{0}, 10^{1}, 10^{2}, 10^{3}\ ] = [0.001, 0.01, 0.1, 1, 10, 100, 1000]$$

Next, I will present all experimental (simulation) results **for applying linear SVMs (no kernel) with L_1 (Lasso) regularization.**

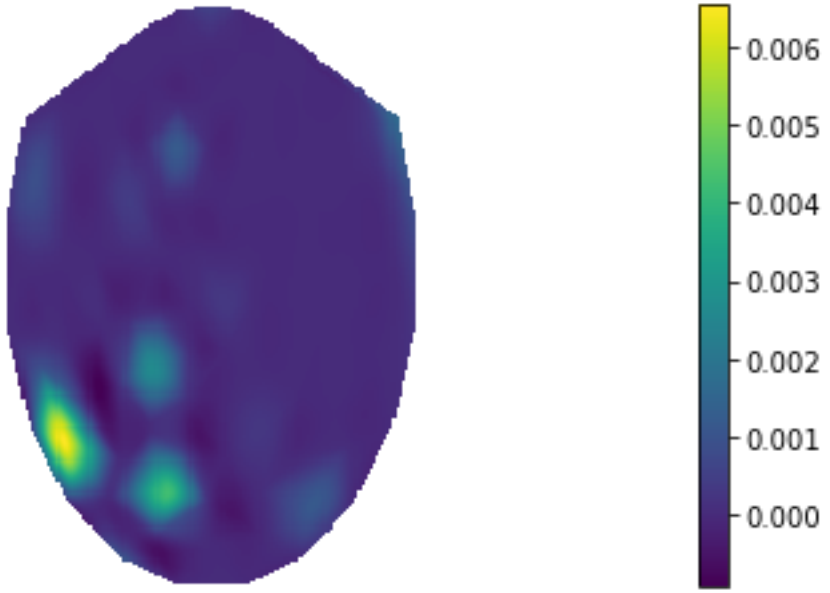
3. Experimental (Simulation) Results

a) For the 1^{st}-level cross-validation fold #3

i. Visualize the weights on the brain surface (using the provided $show\_chanWeight$ function)



Imagined Movement (Linear SVM, Lasso)



Overt Movement (Linear SVM, Lasso)

Key take-away points: The overt movement has higher contrast between the dominant area and other non-dominant area than the imagined movement. We can also see that many weights in the non-dominant area are equal to 0 for both the imagined and overt movement, so the weights are sparse.

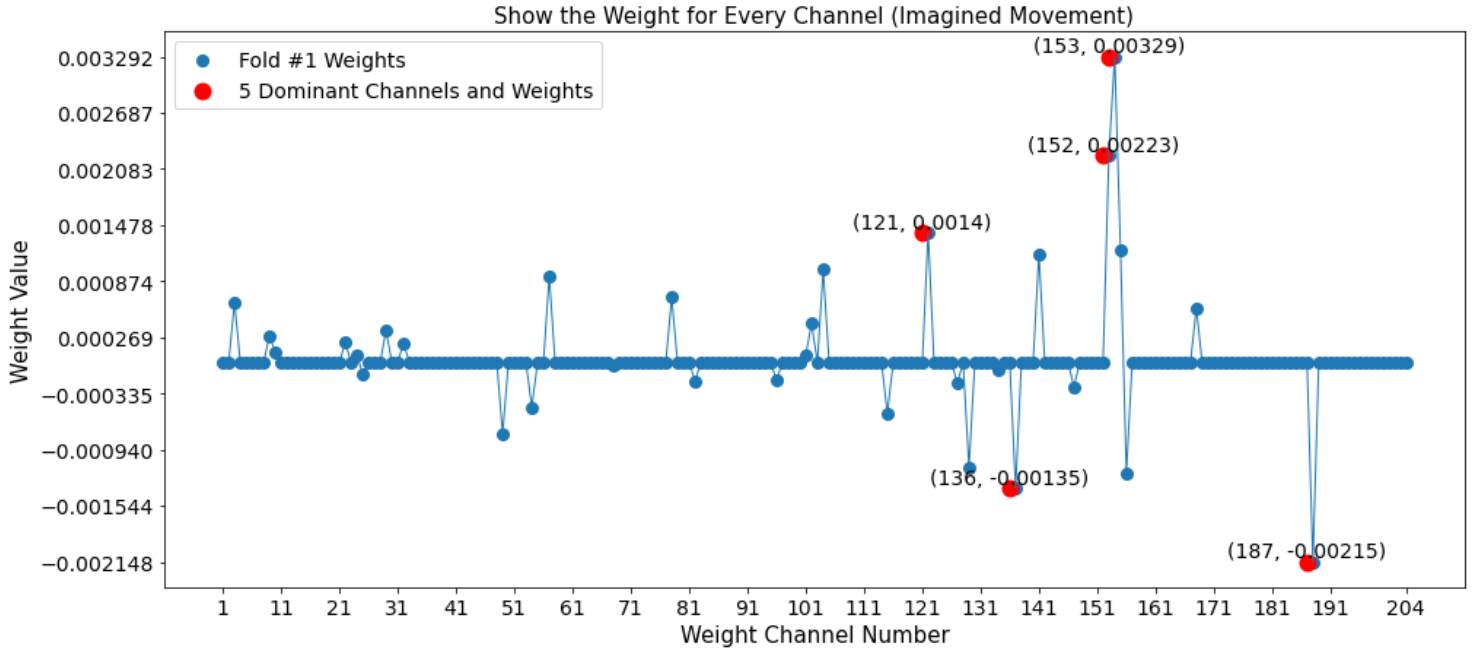
3. Experimental (Simulation) Results

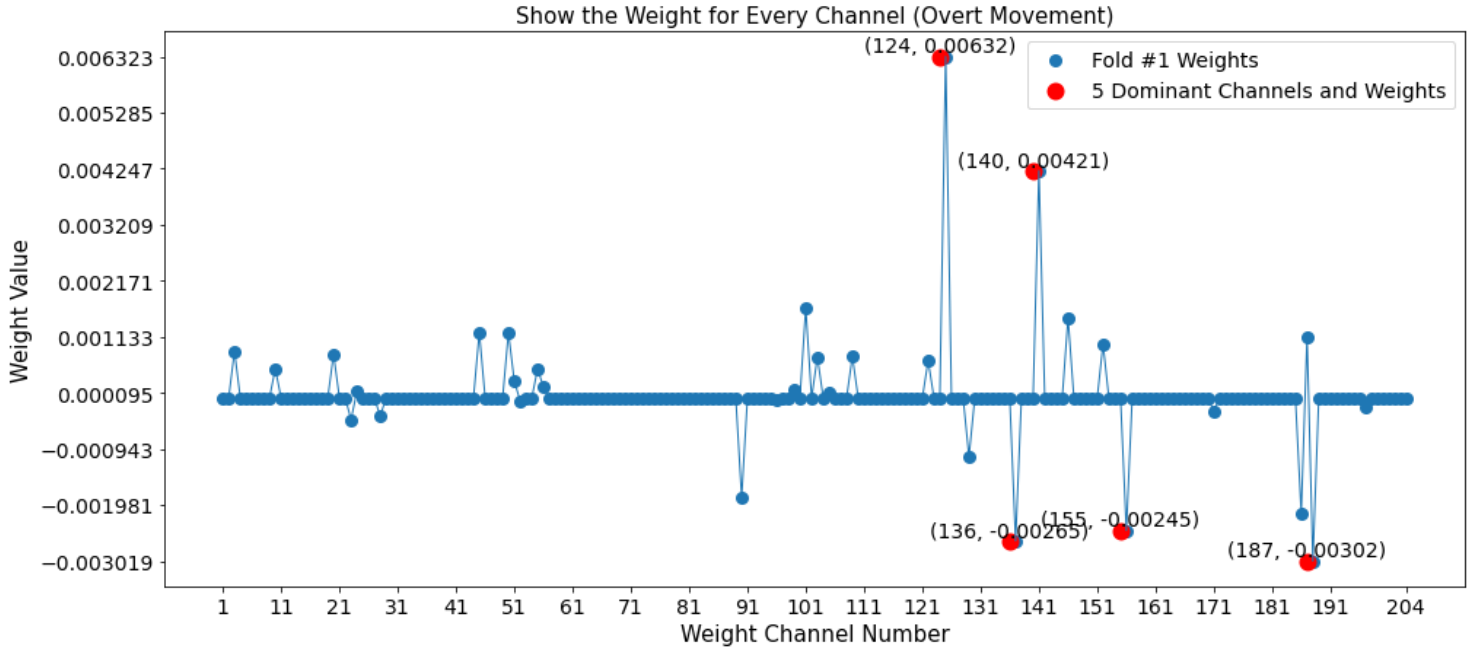
a) For the 1^{st}-level cross validation fold #3

ii. Show (plot) the weight for every channel

iii. Provide a list of the 5 dominant channels and their weights

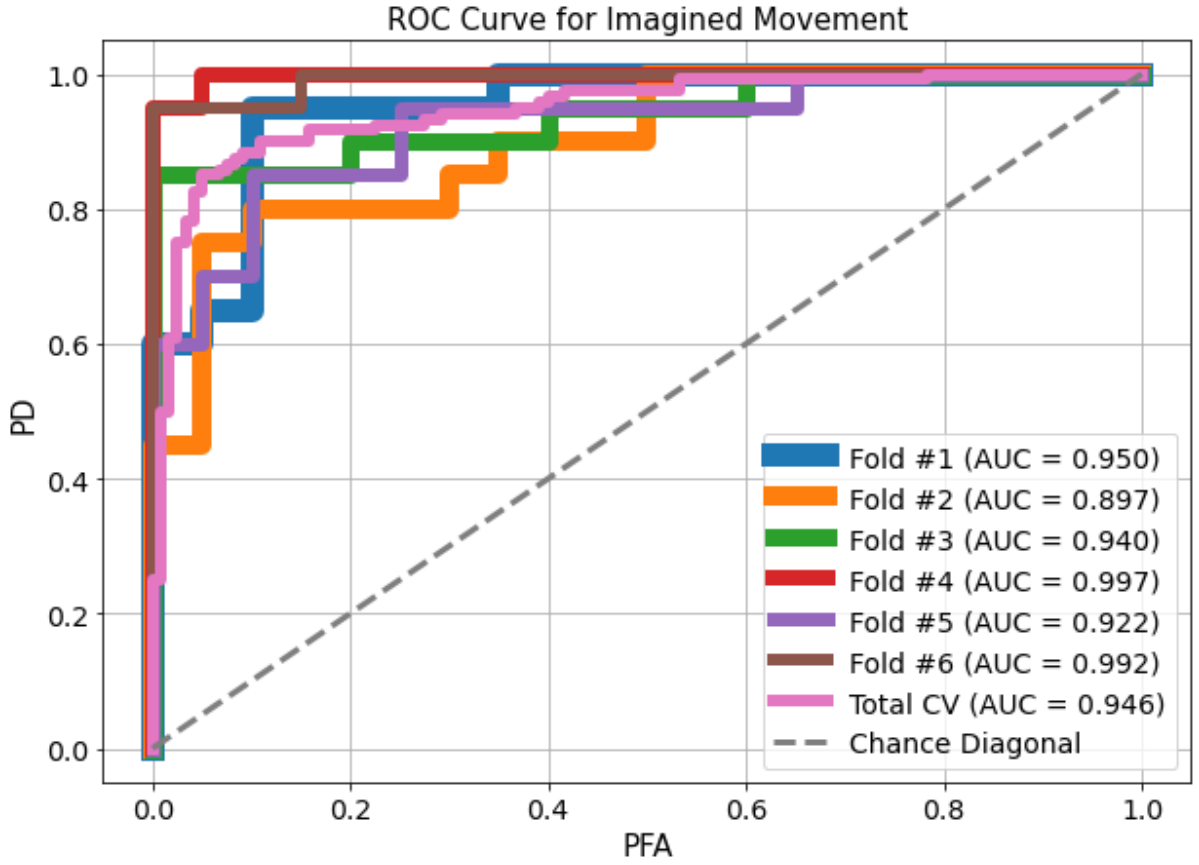
Key take-away points: The overt movement has a larger difference between the 5 dominant weights and other weights than the imagined movement. For both the imagined and overt movement, many weights are exactly equal to 0, so the weights are very sparse. The high sparsity is what we want by applying Lasso regularization.



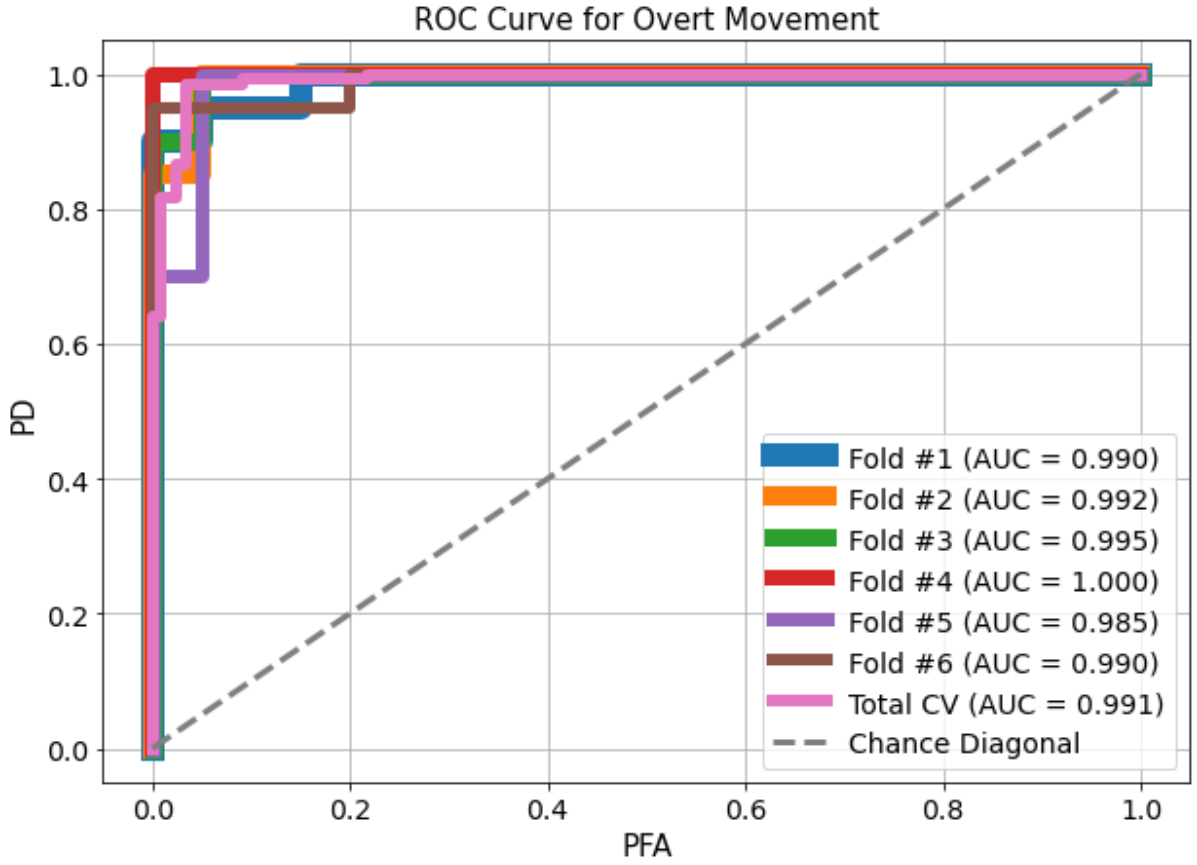


3. Experimental (Simulation) Results

b) Provide the ROC for each 1^{st}-level cross-validation fold (6 ROCs), and the total cross-validated (CV) ROC on a single graph



Imagined Movement (Linear SVM, Lasso)



Overt Movement (Linear SVM, Lasso)

Key take-away points: For the imagined movement, the individual fold ROCs are not representative of the total ROC.
For the overt movement, the individual fold ROCs are representative of the total ROC. In addition, the overt movement has a larger AUC value for the total cross-validated ROC than the imagined movement, which means we can achieve better classification performance with the overt movement.

3. Experimental (Simulation) Results

Provide accuracy for each 1^{st}-level cross-validation fold assuming a threshold $\beta = 0$ (6 accuracy values), and the total cross-validated accuracy. Compare the L_1 (Lasso) regularization parameter values across the two data sets.

	Accuracy	Optimal Lasso Regularization Parameter
Fold #1	0.875	0.01
Fold #2	0.85	10
Fold #3	0.875	1000
Fold #4	0.975	1000
Fold #5	0.825	0.1
Fold #6	0.975	10
Total Cross-validated	**0.8958**	

Imagined Movement (Linear SVM, Lasso)

	Accuracy	Optimal Lasso Regularization Parameter
Fold #1	0.925	0.01
Fold #2	0.95	0.1
Fold #3	0.975	100
Fold #4	0.975	100
Fold #5	0.875	0.01
Fold #6	0.975	10
Total Cross-validated	**0.9458**	

Overt Movement (Linear SVM, Lasso)

Key take-away points: The imagined movement has relatively larger optimized Lasso regularization parameter for individual folds than the overt movement. More importantly, by applying Lasso regularization, we can achieve better total cross-validated accuracy for both the imagined and overt movement than Ridge regularization (0.8792 and 0.9417 respectively).

3. Experimental (Simulation) Results

To earn a 9.5 on this section, after presenting all experimental results of applying **Linear** SVMs (both Ridge and Lasso regularizations have been considered), I also want to introduce **Nonlinearity** to see if I can get even better classification performance. Therefore, I decide to apply a nonlinear SVM with a radial basis function (RBF) kernel to both the imagined and overt data sets.

To implement the second-level cross-validation, here I also consider a large range of L_2 (Ridge) regularization parameters to optimize, these candidate regularization parameters are

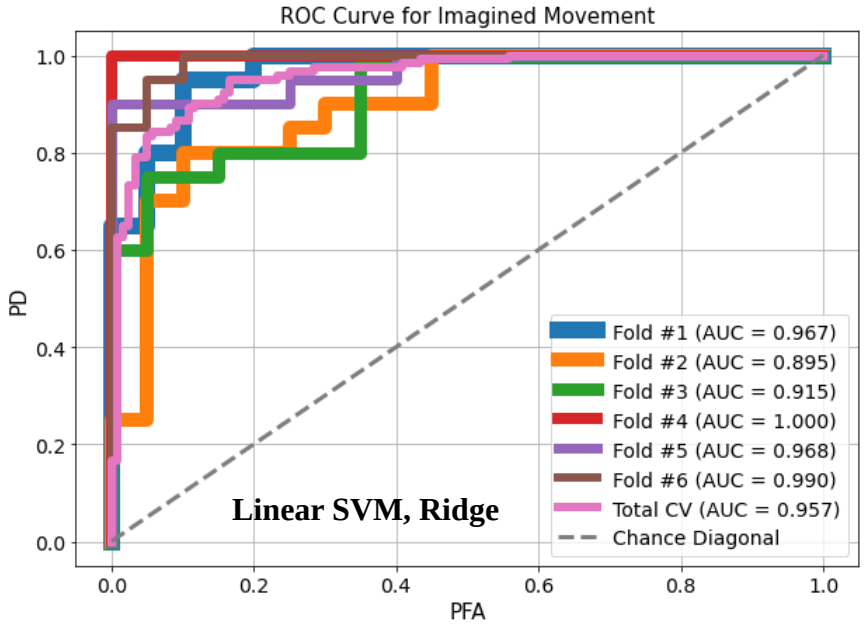
$$\lambda = [10^{-6}, 10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 10^{0}, 10^{1}, 10^{2}, 10^{3}, 10^{4}, 10^{5}, 10^{6}]$$

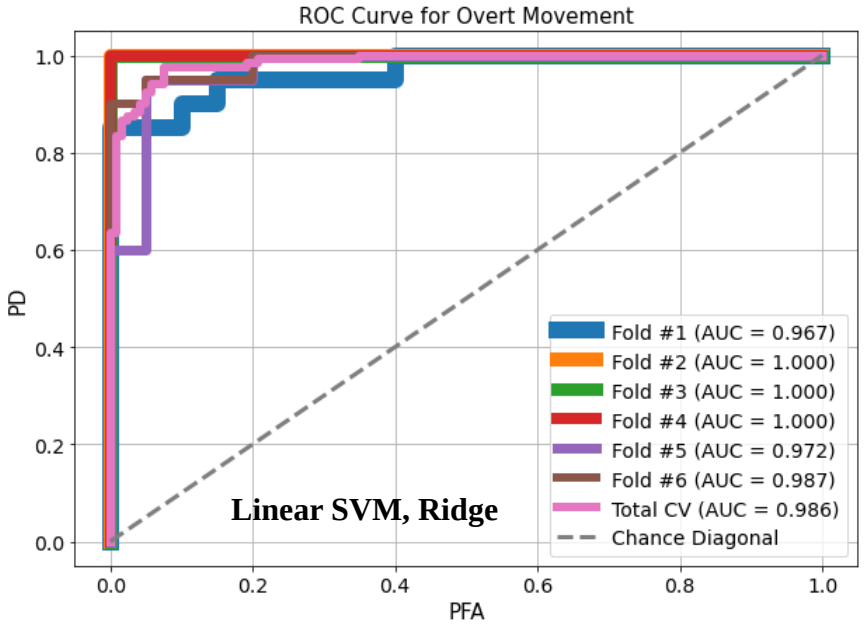
Next, I will present all experimental (simulation) results of applying an SVM with an RBF kernel, and I will compare the classification performance between a nonlinear SVM with an RBF kernel and a linear SVM with L_2 (Ridge) regularization.

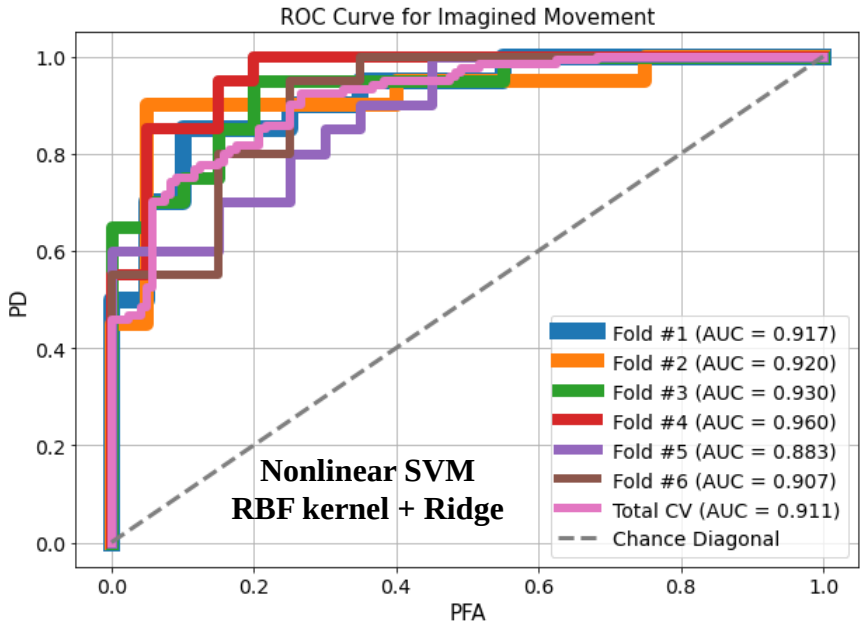
3. Experimental Results

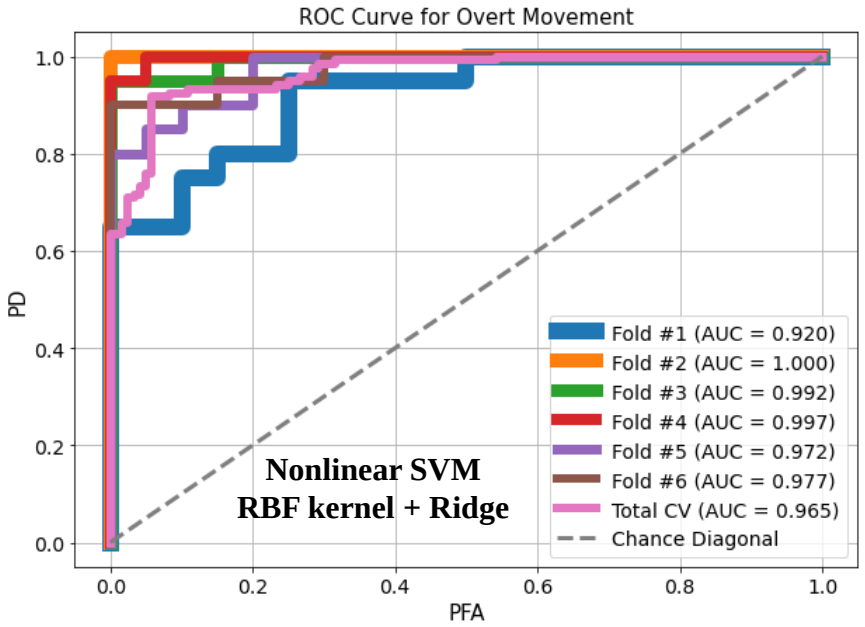
Provide the ROC for each 1^{st}-level cross-validation fold (6 ROCs), and the total cross-validated (CV) ROC on a single graph

Key take-away points: For both the imagined and overt movement, the nonlinear SVM with an RBF kernel has a smaller total cross-validated AUC value than the linear SVM with Ridge regularization, which means that the linear SVM has better classification performance than the nonlinear SVM with an RBF kernel for both the imagined and overt data sets.









3. Experimental Results

Provide and compare accuracy for each first-level cross-validation fold assuming a threshold $\beta = 0$ (6 accuracy values), and the total cross-validated accuracy

Provide and compare the number of support vectors for each first-level cross-validation fold

Key take-away points: For both the imagined and overt movement, the nonlinear SVM with an RBF kernel has a smaller total cross-validated accuracy than the linear SVM with Ridge regularization, which means that the linear SVM has better classification performance than the nonlinear SVM with an RBF kernel for both the imagined and overt data sets. In addition, for both the imagined and overt movement, the nonlinear SVM has approximately 3 times more individual fold support vectors than the linear SVM.

Imagined Movement	Accuracy	Number of Support Vectors
Fold #1	0.85	41
Fold #2	0.80	46
Fold #3	0.825	63
Fold #4	0.975	49
Fold #5	0.90	43
Fold #6	0.925	49
Total Cross-validated	0.8792	**Linear SVM Ridge**

Overt Movement	Accuracy	Number of Support Vectors
Fold #1	0.90	33
Fold #2	0.95	44
Fold #3	0.975	38
Fold #4	0.975	39
Fold #5	0.90	34
Fold #6	0.95	36
Total Cross-validated	0.9417	**Linear SVM Ridge**

Imagined Movement	Accuracy	Number of Support Vectors
Fold #1	0.825	128
Fold #2	0.775	131
Fold #3	0.825	136
Fold #4	0.875	154
Fold #5	0.775	129
Fold #6	0.80	153
Total Cross-validated	0.8125	**Nonlinear SVM RBF kernel + Ridge**

Overt Movement	Accuracy	Number of Support Vectors
Fold #1	0.80	99
Fold #2	0.975	103
Fold #3	0.975	113
Fold #4	0.975	115
Fold #5	0.90	114
Fold #6	0.95	200
Total Cross-validated	0.9292	**Nonlinear SVM RBF kernel + Ridge**

4. Discussion / Conclusions

Discussion Question (1) – Why performance (accuracy) is different?

Based on the experimental results, we can see that for all 3 different types of SVM classifiers (linear SVM with Ridge, linear SVM with Lasso, and nonlinear SVM with RBF kernel), the total cross-validated accuracy for the overt data set is at least 5% (at most over 10%) higher than the total cross-validated accuracy for the imagined data set, so I think the performance (accuracy) is different between the imagined movement and the overt movement.

I think the reason is more about the characteristic of data sets. The 240 imagined EEG trials are collected by imaging making (not actually making) a left or right movement, so they represent relatively weaker signals, which are more difficult for training a machine learning model to correctly classify. However, the 240 overt EEG trials are collected by making a left or right movement, so they represent relatively stronger signals, which are less difficult for training a machine learning model to correctly classify.

Therefore, we can achieve relatively better performance (accuracy) for the overt movement data than the imagined movement data.

4. Discussion / Conclusions

Discussion Question (2) – Why the number of support vectors is different?

Based on the experimental results of applying a linear SVM with Ridge, we can see that the imagined movement has approximately 12 more support vectors than the overt movement, which is an explicit difference if the linear SVM only has around 30, 40, or 50 total number of support vectors after training. Therefore, I think the number of support vectors is different between the imagined movement data and the overt movement data.

To explain why, recall that when we are training an SVM, we want support vectors to define boundaries between different classes, so that we can improve classification performance (accuracy). The more support vectors needed, the more difficult (or more struggling) for an SVM to define boundaries between different classes. This corresponds to the fact that the imagined movement has weaker signals, which means it is more difficult to define a boundary between class 1 (left) and class 2 (right), and therefore more support vectors are needed.

4. Discussion / Conclusions

Discussion Question (3) – Why linear SVMs and nonlinear SVMs have different performance?

Based on the experimental results, we can see that for the imagined and overt movement, a linear SVM with Ridge regularization can achieve 0.8792 and 0.9417 total cross-validated accuracy respectively, while a nonlinear SVM with an RBF kernel (and Ridge regularization) can only achieve 0.8125 and 0.9292 total cross-validated accuracy, respectively. Therefore, we can conclude that a linear SVM has better classification performance (accuracy) than a nonlinear SVM with an RBF kernel.

To explain why, we can see that for both imagined and overt data sets, the nonlinear SVM has approximately 3 times more support vectors than the linear SVM, which means given the high-dimensional EEG signals, the nonlinear SVM is having a very difficult time in defining a nonlinear boundary between class 1 (left) and class 2 (right). I think such difficulty is induced by the fact that we have many features for each EEG observation, but we only have a limited number of EEG observations for model training. Taking the curse of dimensionality into consideration, if we can have much more EEG observations than 240, I believe we will be able to explicitly improve the classification performance (accuracy) using a nonlinear SVM with an RBF kernel.

4. Discussion / Conclusions

Conclusion Topic (1) – Factors that may impact classification accuracy?

Based on all previous experimental results and explanations, we can conclude that, the factors that may impact classification accuracy in this mini-project include but are not limited to

(1) Whether the given EEG observations have strong signals or not (imagined and overt movement)

(2) Whether we have a large amount of training observations compared with the number of features or not

(3) Whether we consider a large range/number of candidate regularization parameters or not

(4) Whether we implement L_2 (Ridge) regularization or L_1 (Lasso) regularization

(5) Whether we apply a linear SVM or a nonlinear SVM with an RBF kernel

4. Discussion / Conclusions

Conclusion Topic (2) – Limits or problems with your approach?

Since I have eventually achieved good classification performance (accuracy) for both the imagined and overt data sets, I do not think there are any significant problems/bugs with my approach to implement two-level cross-validation. However, I do think there are 2 explicit limits when I work on the BCI movement decoding project.

Firstly, when I write Python codes to implement two-level cross-validation, I use many “for” loops to extract each fold and to iterate each candidate regularization parameter, because I think this is the most straightforward way to do that. However, Python is not optimized with many “for” loops written in codes, and I think the relatively long runtime of my two-level cross-validation is partially related to this limit.

Secondly, I do not think I have conducted comprehensively ample experiments (simulation) to thoroughly research on possible factors that may impact classification accuracy, and how exactly they affect classification accuracy. For example, when introducing nonlinearity to SVMs, what about trying to optimize the RBF radius parameter in the second-level cross-validation? What about trying different nonlinear kernels such as the polynomial kernel or the sigmoid kernel?

4. Discussion / Conclusions

Conclusion Topic (3) – Possible improvements that can be made?

I think further efforts can be made to improve the 2 limits described in Conclusion Topic (2).

Firstly, I want to refine my code and avoid the abuse of multiple “for” loops. To do that, I should get myself more familiar with vector and matrix operations in Python, which have been efficiently optimized and accelerated. For example, if I can build a two-dimensional array instead of one, the additional one dimension in my data structure might help me save one “for” loop. With less “for” loops embedded in my codes, I might be able to accelerate my experiments (simulation) a bit.

Secondly, maybe I could conduct more experiments/simulations and research more comprehensively and thoroughly on factors that may impact classification accuracy, and how they affect classification accuracy. Future work research questions could include but might not be limited to

(1) Instead of optimizing the Ridge regularization parameter, what about optimizing the RBF radius parameter in the second-level cross-validation when we are using a nonlinear SVM with an RBF kernel?

(2) What about trying other nonlinear kernels such as the polynomial kernel and the sigmoid kernel?

(3) Although achieving good classification performance for both the imagined and overt data sets, I encounter some convergence problems when using a linear SVM with L_1 (Lasso) regularization. I want to study more about these problems and try to address these convergence issues in the future.

4. Discussion / Conclusions

Conclusion Topic (4) – Anything unique you have done to improve/validate your classifier's accuracy/efficiency?

As presented in experimental results and explained in Discussion Question (3), I think there are 2 unique things I have done trying to improve my classifier's accuracy.

Firstly, in addition to applying a linear SVM with L_2 (Ridge) regularization to meet the minimum experimental requirements of this mini-project, I also apply a linear SVM with L_1 (Lasso) regularization. The experimental results demonstrate that I can achieve better total cross-validated accuracy for both the imagined and overt data sets, compared with Ridge regularization.

Secondly, in addition to applying linear SVMs (Ridge and Lasso), I also apply a nonlinear SVM with an RBF kernel (and Ridge regularization) trying to further improve my classifier's accuracy. However, the experimental results show that introducing nonlinearity does not improve my classifier's accuracy but degrades the total cross-validated accuracy for both the imagined and overt data sets, compared with linear SVMs. My interpretation and analysis for this fact is written in Discussion Question (3).

5. References / Citations

References from books or journal articles

[1] Lotte, F., Congedo, M., Lécuyer, A., Lamarche, F., & Arnaldi, B. (2007). A review of classification algorithms for EEG-based brain–computer interfaces. Journal of neural engineering, 4(2), R1.

[2] Santoso, L., Singh, B., Rajest, S., Regin, R., & Kadhim, K. (2020). A genetic programming approach to binary classification problem. EAI Endorsed Transactions on Energy Web, 8(31), e11.

[3] Wang, X., Yang, Z., Chen, X., & Liu, W. (2019). Distributed inference for linear support vector machine. Journal of machine learning research, 20.

[4] Hanneke, S., & Kontorovich, A. (2021, March). Stable sample compression schemes: New applications and an optimal SVM margin bound. In Algorithmic Learning Theory (pp. 697-721). PMLR.

[5] Bounsiar, A., Beauseroy, P., & Grall, E. (2005, September). A straightforward SVM approach for classification with constraints. In 2005 13th European Signal Processing Conference (pp. 1-4). IEEE.

[6] Wang, J., Liu, J., & Liu, Y. (2021). Improved Learning Rates of a Functional Lasso-type SVM with Sparse Multi-Kernel Representation. Advances in Neural Information Processing Systems, 34.

[7] Yu, Y., McKelvey, T., & Kung, S. Y. (2013, May). A classification scheme for ‘high-dimensional-small-sample-size’data using soda and ridge-SVM with microwave measurement applications. In 2013 IEEE International Conference on Acoustics, Speech and Signal Processing (pp. 3542-3546). IEEE.

[8] Fletcher, R. (1971). A general quadratic programming algorithm. IMA Journal of Applied Mathematics, 7(1), 76-91.

[9] Köppen, M. (2000, September). The curse of dimensionality. In 5th online world conference on soft computing in industrial applications (WSC5) (Vol. 1, pp. 4-8).

5. References / Citations

Provide a citation for every toolbox or package I leverage when coding

I use Python to work on this project. All leveraged packages and the correspondingly leveraged purposes are described below.

(1) I leverage the “show_chanWeight” function to visualize the weights on the brain surface. This function belongs to starter code provided with the project description and information.

(2) I do not use “SciPy” package to do my own coding, but I notice that the provided “show_chanWeight” function does import this package, therefore I also provide the “SciPy” package as a citation.

(3) I leverage “pandas” package to read the provided imagined and overt data sets.

(4) I leverage “NumPy” package to process multi-dimensional array/matrix.

(5) I leverage “matplotlib” package to plot the weight for every channel and to plot the ROC curve.

(6) I leverage “scikit-learn” package to implement two-level cross-validation, to implement a linear SVM with Ridge regularization, to implement a linear SVM with Lasso regularization, to implement a nonlinear SVM with an RBF kernel (and Ridge regularization), and to calculate the area under curve (AUC) value.

6. Collaboration Descriptions

I only shared and debated some **general/conceptual** ides such as two-level cross-validation and Ridge regularization with my Peer Feedback team members during 2 Peer Feedback Sessions held in class. These team members are: Fakrul Tushar, Haocheng Meng, Aditya Bhamidipati, and Aniket Mukherjee. **Apart from the 2 Peer Feedback Sessions held in class, I did not collaborate with anyone. This statement means that**

(1) I did not share code with anyone while working on this project.

(2) No one shared code with me while working on this project.

(3) I did not compare results with anyone while working on this project.

(4) I did not help anyone overcome an obstacle while working on this project.

(5) No one helped me overcome an obstacle while working on this project.