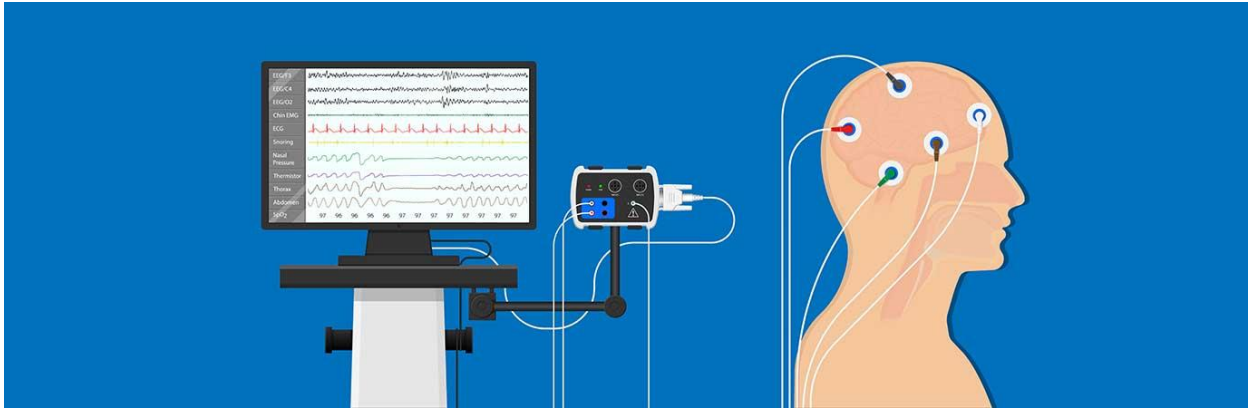
Brain-Computer Interface Movement Decoding

Ian Hanus

Background

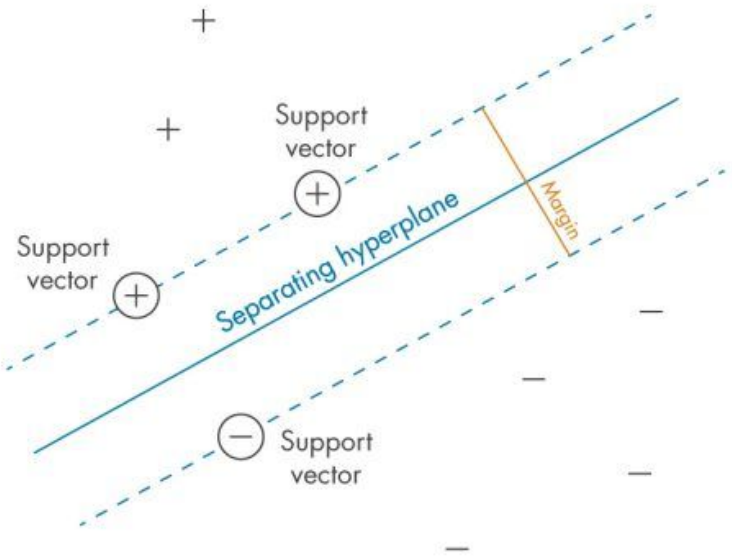
Brain computer interfaces (BCIs) are often used as a communication option for people with neuromuscular impairments that don't allow them to use interfaces such as keyboards. A BCI allows the user to interact with a system using electroencephalographic (EEG) activity monitors, which record the electrical activity on the scalp as a proxy for activity of the brain underneath. This is an important step in increasing accessibility to technology, lowering the barrier for entry into many fields for people with physical impairments. It is, then, extremely important that a BCI operate quickly and accurately to improve accessibility and equality for those with disabilities.



Brain computer interface
https://www.meta.org/feed-previews/brain-computer-interface/dc5bbc99-fe4c-46b8-a130-a409bcf1dc10

Project Goal

The main goal of this project is to create an algorithm that can differentiate between the “left” and “right” directions given data obtained using EEG activity monitors. Throughout this process, we will examine the efficacy of support vector machines (SVMs) in separating these two classes, as well as the effect of LASSO and Ridge regularization of the data.

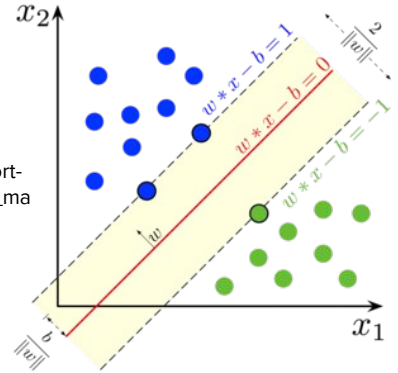


Defining the “margin” between classes – the criterion that SVMs seek to optimize.
https://www.mathworks.com/discovery/support-vector-machine.html

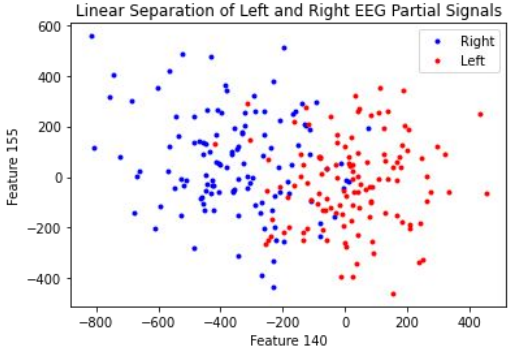
Mathematical Formulation (SVMs)

Support vector machines are responsible for finding a decision boundary between classes that maximizes the margin between the classes. A 2D example is shown to the right. While there are an infinite number of lines that could separate the blue and green classes correctly, there is an optimal line that maximizes the margin between the decision boundary and the points closest to the decision boundary, resulting in a. In higher dimensional data, the same can be done for a hyperplane. Because we are dealing with data characterized by 204 features, we will be looking for a 203-dimension hyperplane to separate the data.

SVMs are a particularly good choice for this problem because of how linearly separable the data is. The figure to the right shows the split of the “left” and “right” data in the feature space consisting of the 140th and 155th features. The linear nature of the split makes it seem likely that a linear SVM classifier will perform well, especially with many more features providing possibly useful information.



Maximum-margin hyperplane and margin for an SVM trained on two classes https://en.wikipedia.org/wiki/Support-vector_machine#/media/File:SVM_margin.png



Separation of “right” and “left” classes in the 140/155 feature space

Mathematical Formulation (cont.)

As stated in the previous slide, SVMs operate by maximizing the perpendicular distance between the decision boundary and the nearest observation. The distance from the observation to the decision boundary, or the margin, is equal to

$$y(x)=w^{T}x+b$$

The classes that you are trying to separate are each designated -1, and 1, as a class. These classes are chosen to ensure that all observations of a certain class can be classified as the correct class through sign alone. To ensure all training observations are the right sign, we set

$$y(x)=c(w^{T}x+b)$$

where c is the class of the training point, either -1 or 1. This means that y will be positive for all points x. To normalize the distance from the decision boundary to the margin, and help us later in optimization using Lagrange multipliers

$$y(x)=c(w^{T}x+b)/||w||$$

Mathematical Formulation (cont.)

Now we have established the function that we are trying to optimize: as we are concerned with the closest points, we are trying to find the weights ω and the offset b that will maximize the minimum distance between the decision boundary and any of the data points. So the function that we will optimize will be:

$$f(x)=\text{argmax}_{w,b}\{\min_t[c_t(w^T x_t+b)]/||w||\}$$

The next question is how to optimize this: the answer is using Lagrange multipliers! Because we normalized by dividing by ||w||, we know that the distance from the decision boundary to the margin will be 1. Therefore, we know that the distance to the margin cannot be less than one for any point, so we can apply the constraint

$$c_t(w^T x_t+b) \geq 1 \ \forall \ x_t$$

and so the Lagrangian problem we would be solving is

$$L(w, b) = 1/2||w||^2 - \Sigma_{t=1 \to N} k_t\{c_t(w^T x_t+b)-1\}$$

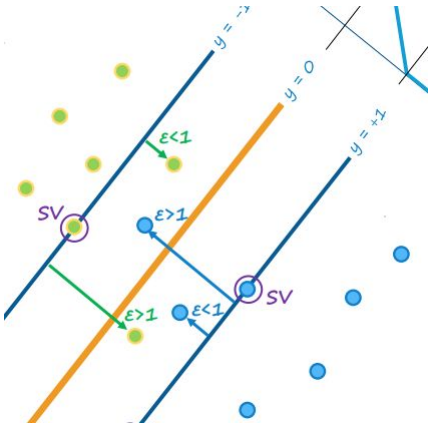
where k are the Lagrangian coefficients.

Mathematical Formulation (cont.)

Solving for the previous Lagrangian would give you a hard margin: it would perform extremely well on the training data assuming it is linearly separable, but would overfit. To avoid overfitting and make allowances for non-perfectly linear data we need to add in a slack variable ε. This variable will be 0 if the point is classified correctly outside of the support boundary, a value between 0 and 1 if is between the support boundary and the decision boundary on the correct side, and a value greater than one if it is on the wrong side of the decision boundary. Our constraint can then be changed to include the slack variable such that

$$c_t(w^T x_t + b) \geq 1 - \varepsilon_t \ \forall \ x_t$$

Because of how we defined the slack variable, we now also have the constraint $\varepsilon_t \geq 0 \ \forall \ t$.



Slack Variable
Tantum SVM slides (24)

Mathematical Formulation (cont.)

This means that the Lagrangian can now be rewritten with the new constraints as

$$L(w, b, k, \mu) = 1/2||w||^2 - \Sigma_{t=1\to N} k_t\{c_t(w^T x_t+b)-1 + \varepsilon_t\} - \Sigma_{t=1\to N} \mu_t \varepsilon_t + C\Sigma_{t=1\to N} \varepsilon_t$$

The final term, $C\Sigma_{t=1\to N} \varepsilon_t$, accounts for regularization in the model. As C goes up, regularization goes down, acting as the inverse of the regularization parameter λ that is often used as a parameter that increases regularization as it increases. Solving this using the method of Lagrange multipliers will result in the SVM we will be using for the project, using the previously listed constraints of $c_t(w^T x_t+b) \geq 1 - \varepsilon_t \ \forall \ x_t$ and $\varepsilon_t \geq 0 \ \forall$ t.

Mathematical Formulation (cont.)

The regularization we will be using for this project can also take two forms: L1 or L2 penalty, which will now be referred to as LASSO and Ridge regularization respectively. LASSO stands for least absolute shrinkage and selection operator, and it only takes the magnitudes of the weight vector into account. This means that it will shrink small weights to zero, resulting in a more sparse feature space. The cost function for LASSO is

$$\sum_{i=1}^{M} (y_i - \hat{y}_i)^2 = \sum_{i=1}^{M} \left(y_i - \sum_{j=0}^{p} w_j \times x_{ij} \right)^2 + \lambda \sum_{j=0}^{p} |w_j|$$

Ridge regularization uses a cost function that adds a penalty equivalent to the square of the magnitude of the coefficients.

$$\sum_{i=1}^{M} (y_i - \hat{y}_i)^2 = \sum_{i=1}^{M} \left(y_i - \sum_{j=0}^{p} w_j \times x_{ij} \right)^2 + \lambda \sum_{j=0}^{p} w_j^2$$

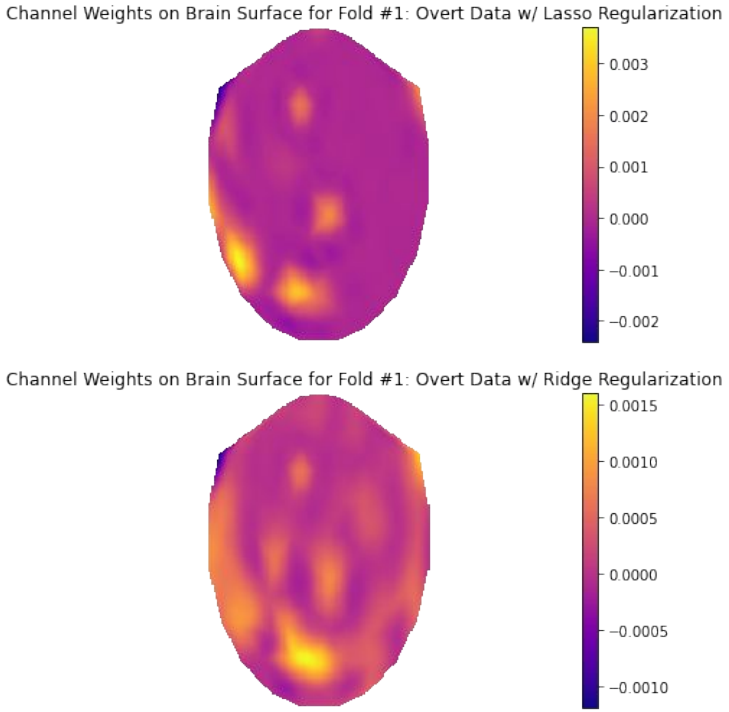
This shrinks coefficients, but does not push any of the feature weights to zero.

Experimental Results

The following results were obtained using 2 datasets with balanced “right” and “left” true signals for a total of 240 signals each, one with the signals being obtained through overt motions and the other being obtained through imagined motions. The experimental results were obtained using 2-level cross validation, where the first level consisted of 6 sets of 40 observations (20 “left” and 20 “right”). The second level of cross validation was used to tune the hyperparameter C, which controlled the amount of regularization applied to the support vector machine weights. Once the hyperparameter was optimized on the second level, the optimal C value was applied to the first level and common cross-validation was used to find the probability of correct decision for each of the folds as well as for the overall dataset.

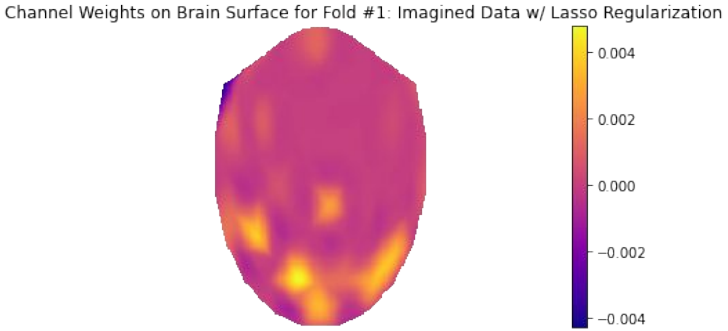
Overt Data Channel Weights: On Brain

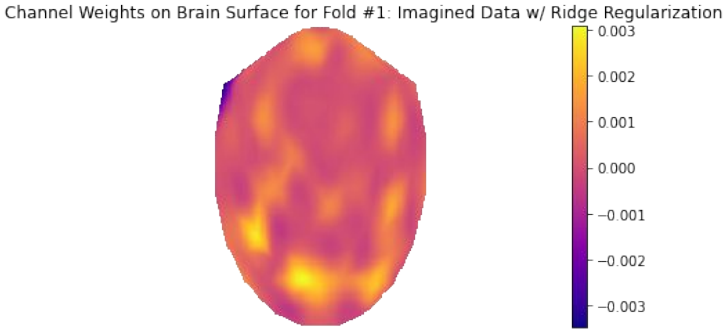
To the right, we see that overt data tended to weight channels towards the back left and back center as more important, with some spotty importance around the front center and front right as well. As we expected, LASSO regularization resulted in more sparse channel weights, with many fewer bright areas relative to the dark areas.



Imagined Data Channel Weights: On Brain

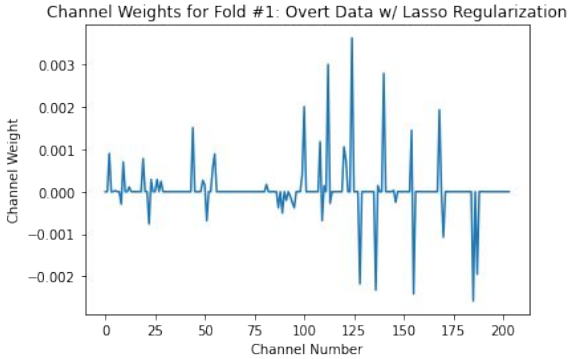
While the imagined weights are similar to the overt weights in that the back left/center is highly weighted, the imagined weights also tend to weigh the signals from the back right of the brain more highly. Once again we see that the LASSO regularization results in more sparse channel weights than the Ridge regularization, which has many more light (or highly weighted) spots.

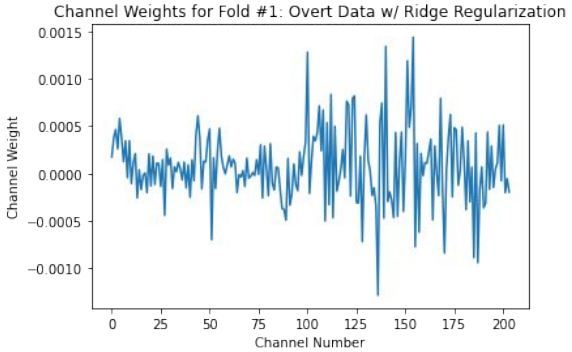




Overt Data Channel Weights

Here we can see the sparsity of the channel weights even more clearly. LASSO regularization pushes the less important channel weights to absolute zero, whereas the Ridge regularization doesn't. The two do, however, both weigh channel number 155 quite highly.





Overt Data w/ Lasso Regularization

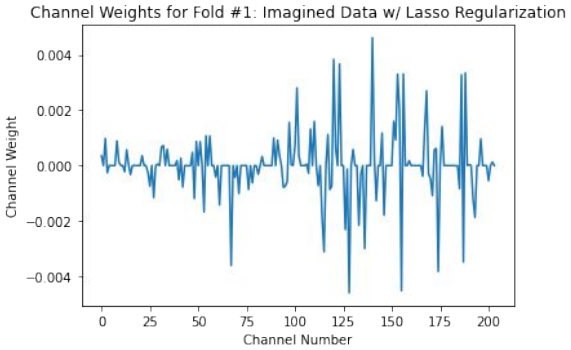
Channel Number	Weight
124	0.00363
112	0.00301
144	0.00279
155	0.00258
185	-0.00249

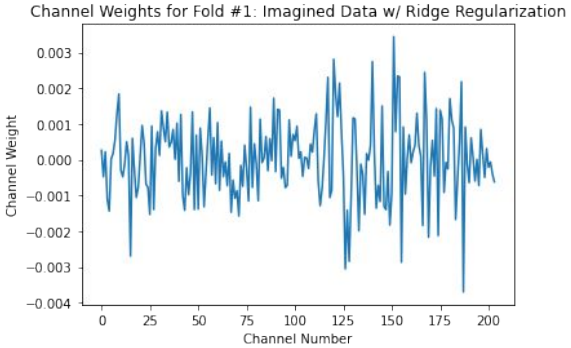
Overt Data w/ Ridge Regularization

Channel Number	Weight
155	0.00144
136	-0.00134
140	0.00128
151	0.00128
100	0.00119

Imagined Data Channel Weights

Although the imagined model w/ LASSO regularization is less sparse than the overt model w/ LASSO regularization, it is still much more sparse than the imagined model w/ Ridge regularization. Here we see that the two models both weight feature number 120 quite highly, and actually both weigh 155 highly as well, just like the overt models did.





Imagined Data w/ Lasso Regularization

Channel Number	Weight
140	0.00462
128	-0.00460
155	-0.00451
120	0.00383
185	-0.00383

Imagined Data w/ Ridge Regularization

Channel Number	Weight
187	-0.00370
151	0.00344
126	-0.00306
120	0.00286
155	0.00284

Overt Accuracies & ROCs

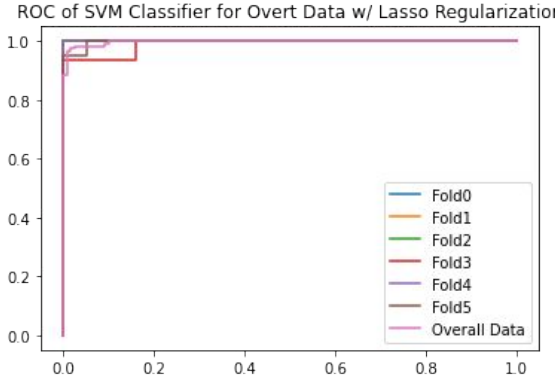
Both the LASSO and Ridge regularization models produced ROCs and accuracies for each of their folds that were representative of the overall ROC and accuracy for the whole overt dataset. While both performed relatively well, the LASSO regularized model consistently outperformed the Ridge regularized model.

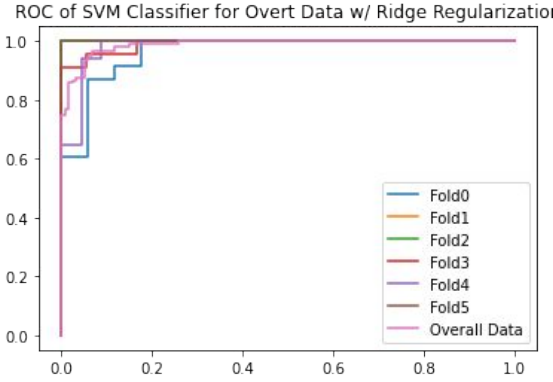
Accuracy of Overt Data w/ Lasso Regularization

	Accuracy
Fold 1	0.975
Fold 2	0.950
Fold 3	1.000
Fold 4	0.975
Fold 5	1.000
Fold 6	0.975
Overall	0.979

Accuracy of Overt Data w/ Ridge Regularization

	Accuracy
Fold 1	0.900
Fold 2	0.925
Fold 3	1.000
Fold 4	0.925
Fold 5	0.950
Fold 6	0.975
Overall	0.946





Imagined Accuracies & ROCs

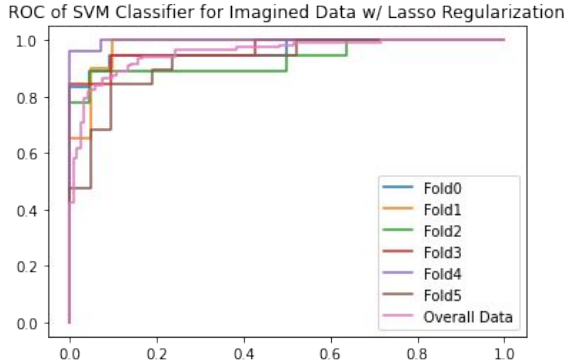
While each of the ROCs of the folds look somewhat representative of the overall ROC for both LASSO and Ridge regularization, and the individual fold accuracies are representative of the overall accuracy for LASSO regularization, the 6th fold of the Ridge regularization is somewhat of an outlier.

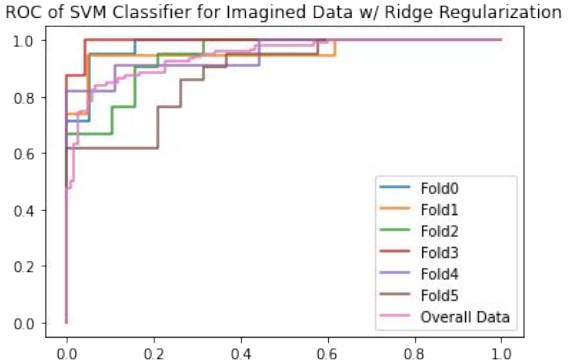
Accuracy of Imagined Data w/ Lasso Regularization

	Accuracy
Fold 1	0.925
Fold 2	0.900
Fold 3	0.925
Fold 4	0.925
Fold 5	0.850
Fold 6	0.850
Overall	0.895

Accuracy of Imagined Data w/ Ridge Regularization

	Accuracy
Fold 1	0.900
Fold 2	0.925
Fold 3	0.875
Fold 4	0.950
Fold 5	0.900
Fold 6	0.750
Overall	0.888





Regularization Parameters

The regularization parameter that we are looking at here is λ, the inverse of the C previously described in methods. The higher the λ, the more regularization occurs for the weights. The overt has a higher regularization value than the imagined for LASSO regularized data, but the inverse is true for the Ridge models. The Imagined Ridge model has regularization parameter values at both ends of the observed range, tending towards both the very large and very small extremes.

Regularization Parameters

	Overt Lasso λ	Overt Ridge λ	Imagined Lasso λ	Imagined Ridge λ
Fold 1	1	0.01	0.000001	0.0001
Fold 2	0.000001	1	100	10000000
Fold 3	1	10000	0.00001	10000000
Fold 4	100	10000	0.00001	10000000
Fold 5	100	10000	1	10000000
Fold 6	1	10000	0.000001	0.000001

Number of Support Vectors

The number of support vectors used for classifying the overt data is less than the number of support vectors used to classify the imagined data for both LASSO and Ridge regularization. However, while all of the other models have an average of 25-40 support vectors, the imagined ridge model had an average of 107.8 support vectors. This number varied over multiple runs from an average of 78.5 to 120.0, but was consistently much higher than the other models.

Number of Support Vectors

	Overt Lasso λ	Overt Ridge λ	Imagined Lasso λ	Imagined Ridge λ
Fold 1	34	18	30	130
Fold 2	13	36	37	137
Fold 3	27	101	32	53
Fold 4	30	13	30	137
Fold 5	13	43	34	56
Fold 6	37	29	33	134
Average	25.6	40.0	32.6	107.8

Discussion

Overall, we saw that our models performed better on overt data than on imagined data regardless of regularization type. This is to be expected, as the signal to noise ratio (SNR) is higher for actual movement than imagined movement: the actual action produced a stronger signal than imagining the action. This could also be a major reason why the regularization parameters for the LASSO models were higher for the model trained on the overt data. To regularize an equivalent amount, with data that have higher values, the regularization parameter λ had to be higher.

Probability of Correct Decision

	Average Accuracy
Overt Lasso	0.979
Overt Ridge	0.948
Imagined Lasso	0.895
Imagined Ridge	0.888

Discussion (cont.)

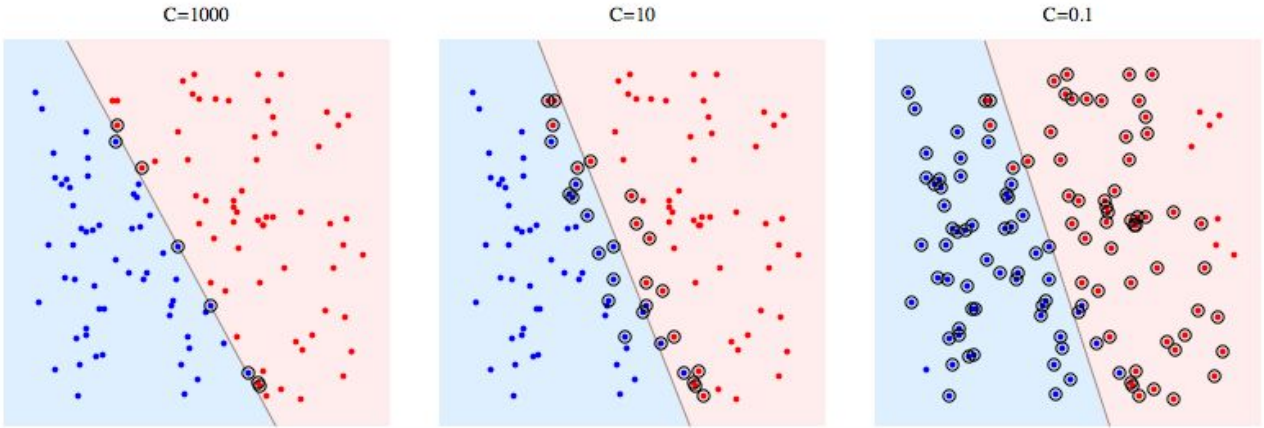
One worrying aspect about these results is the strange values of the regularization parameter recorded for the model trained on the imagined data with the Ridge regularization. This model performed the worst, with the lowest overall probability of correct decision as well. This could possibly be because of overfitting: while SVMs are generally considered somewhat resistant to over-fitting, this is often because of the tuning of the regularization parameter. Because of the extremely high and low magnitude regularization parameters, the model could be overfitting as a result of poor regularization parameters.

Regularization Parameters

	Overt Lasso λ	Overt Ridge λ	Imagined Lasso λ	Imagined Ridge λ
Fold 1	1	0.01	0.000001	0.0001
Fold 2	0.000001	1	100	10000000
Fold 3	1	10000	0.00001	10000000
Fold 4	100	10000	0.00001	10000000
Fold 5	100	10000	1	10000000
Fold 6	1	10000	0.000001	0.000001

Discussion (cont.)

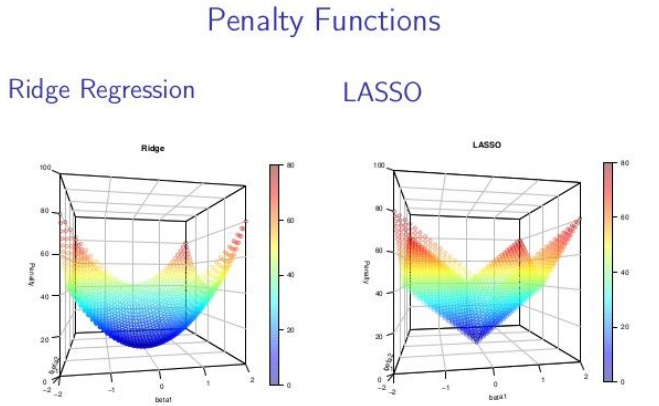
The number of support vectors recorded for the SVM trained on imagined data with Ridge regularization was also worrying. However, this is controlled by the regularization parameter. The higher the regularization parameter C, which is the inverse of λ, the fewer support vectors there will be. That means for the high values of λ that were recorded for this model (or the low values of C), there would be many more support vectors.



Effect of C on Number of Support Vectors
https://stackoverflow.com/questions/4629505/svm-hard-or-soft-margins/4630731

Discussion (cont.)

As an extension, I chose to compare the effects of LASSO and Ridge regularization. Somewhat surprisingly, LASSO performed better on both the imagined and overt data. The major difference between the two is that LASSO has the ability to completely zero out a features ability to influence a model. As there was no reason to believe that any of the features were harmful, I assumed that Ridge regularization leaving all features in would result in more information and a higher classification accuracy. However, LASSO did outperform Ridge across the board. This could be because some features given non-zero weights for the Ridge models were actually somewhat confounding. This means that, in the feature spaces that LASSO zeroed, the Ridge models actually predicted the wrong class for a testing observation based on the fed training data. Because LASSO had effectively ignored this confounding feature, it did not affect the classification of the LASSO model.



Visual Explanation of Ridge Regression and LASSO
https://www.slideshare.net/kaz_yos/visual-explanation-of-ridge-regression-and-lasso

Conclusion

To tie this back to the goal of the project, we must more carefully consider performance on the imagined dataset. While performance on the overt dataset is interesting, it's important to realize that people with certain neuromuscular impairments may not be able to make any overt movements whatsoever. Because of the higher probability of correct decision, I would recommend using the LASSO regularization model if one must be chosen, but with the low probability of correct decision of 0.895 I think that its use cases are limited. For someone who has to interact with this system constantly, a system that misclassifies a simple instruction 10% of the time would be not only extremely aggravating to use, but depending on the application could also be harmful (i.e. aided movement). In conclusion, while this is an interesting experiment, it does not seem to have many realistic use cases until the probability of correct decision is improved.

References

- Chamberlain, Andrew, Ph.D. “A Simple Explanation of Why Lagrange Multipliers Works.” Medium, Medium, 21 Mar. 2017, medium.com/@andrew.chamberlain/a-simple-explanation-of-why-lagrange-multipliers-works-253e2cdcbf74
- Hastie, Trevor. "Ridge regularization: An essential concept in data science." Technometrics 62.4 (2020): 426-433.
- Ho, Chia-Hua, and Chih-Jen Lin. "Large-scale linear support vector regression." The Journal of Machine Learning Research 13.1 (2012): 3323-3348
- Liao, R. "Support vector machines." CSC, 2015.
- Long, Jinyi, Yuanqing Li, and Zhuliang Yu. "A semi-supervised support vector machine approach for parameter setting in motor imagery-based brain computer interfaces." Cognitive neurodynamics 4.3 (2010): 207-216.

References (cont.)

- Saccoccio, Mattia, et al. "Optimal regularization in distribution of relaxation times applied to electrochemical impedance spectroscopy: ridge and lasso regression methods-a theoretical and experimental study." Electrochimica Acta 147 (2014): 470-482.
- Staelin, Carl. "Parameter selection for support vector machines." Hewlett-Packard Company, Tech. Rep. HPL-2002-354R1 1 (2003).
- Vallabhaneni, Anirudh, Tao Wang, and Bin He. "Brain—computer interface." Neural engineering. Springer, Boston, MA, 2005. 85-121.
- Wolpaw, Jonathan R., et al. "Brain-computer interface technology: a review of the first international meeting." IEEE transactions on rehabilitation engineering 8.2 (2000): 164-173.

Python Packages Used:

sklearn.svm.SVC, numpy, pandas, matplotlib, cipy, sklearn.pipeline, sklearn.model_selection, sklearn.metrics