BRAIN-COMPUTER INTERFACE MOVEMENT DECODING

ECE 580 Mini Project #2

4/27/2022 ***Tyler Feldman***

Table of Contents

Introduction

Project Description

In this project, **we constructed a machine learning model to predict what selection (left vs. right) a person would like to make just based on electroencephalogram (EEG) signals**. EEG devices read electrical potential from the brain using electrodes that can be placed on the scalp[1]. These electrodes can be placed on different lobes on the brain to monitor the electrical activity across different locations. From medical science, it is established that specific actions, activity, and states are controlled by specific locations in the brain[1]. Studies have found that left and right lower limb movement can be differentiated using EEG signals[2], and we will try applying this idea to our dataset with a machine learning model.

The goal of this project is first creating a model that can accurately predict based on their EEG signals if someone would like to select a left or right option presented to them. This is a step to creating advanced brain-computer interfaces (BCIs), which act as a communication link between the brain and a computer or other device. Their applications range from rehabilitation, affective computing, neuroscience, and robotics[3].

To train and evaluate our model, we have two different datasets. The first dataset, which we call the "overt " dataset, has EEG signals from subjects were physically moving either their left or right hands. The "imagined" dataset has subjects imagining moving their left or right hands, but not physically moving them.

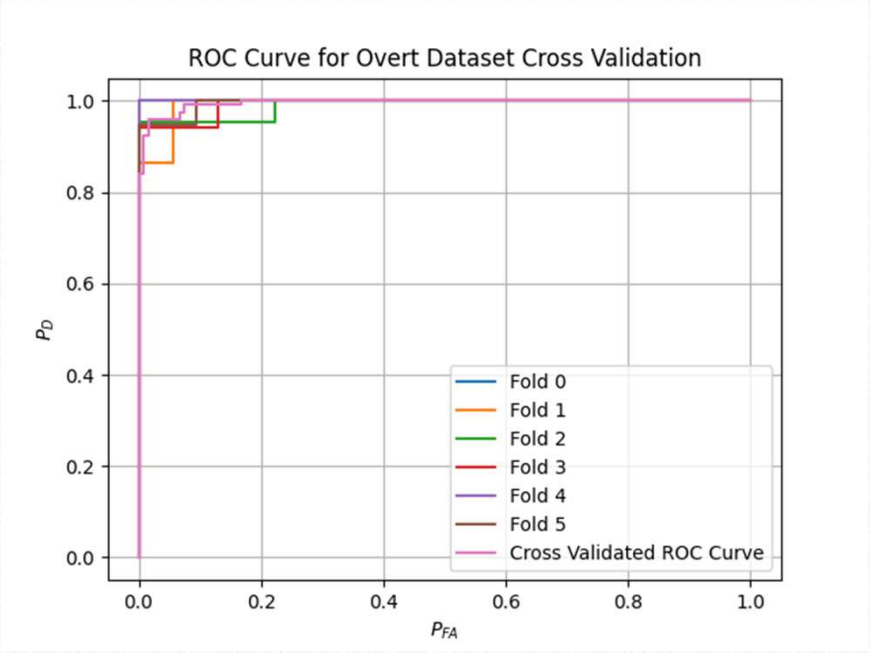
1. Kumar, J. Satheesh, and P. Bhuvaneswari. "Analysis of Electroencephalography (EEG) Signals and Its Categorization–A Study." *Procedia Engineering*, vol. 38, 2012, pp. 2525–2536., https://doi.org/10.1016/j.proeng.2012.06.298.
2. Kline, Adrienne, et al. "EEG Differentiates Left and Right Imagined Lower Limb Movement." *Gait & Posture*, vol. 84, Feb. 2021, pp. 148–154., https://doi.org/10.1016/j.gaitpost.2020.11.014.
3. Saha, Simanto, et al. "Progress in Brain Computer Interface: Challenges and Opportunities." *Frontiers in Systems Neuroscience*, vol. 15, 25 Feb. 2021, https://doi.org/10.3389/fnsys.2021.578875.

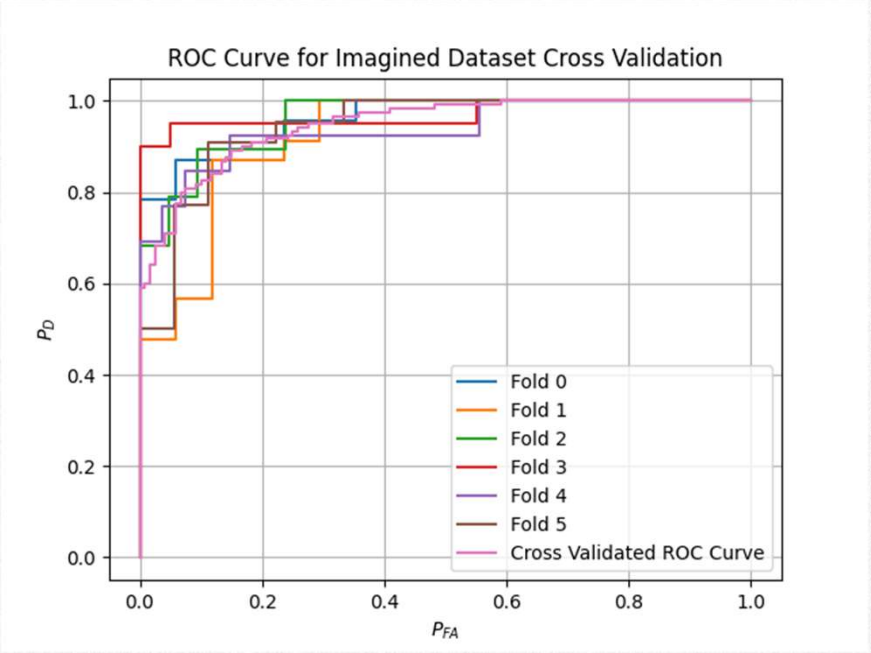
Summary of Key Results

For our machine learning model, I used a support vector machine (SVM) to classify left vs right movement. **The classification model performed extremely well for the overt data, while performing not quite as well for the imagined dataset**. This can be seen by the average cross-validated accuracy in the table below.

Dataset	Accuracy
Overt	0.954
Imagined	0.863

Below are the ROC curves for the model on the overt dataset and for the imagined dataset. The further the graph reaches the top left, the better, so we can see that the ROC for the overt dataset is better than imagined.





Mathematical Formulation

Formulation of the Classification Problem

The dataset we are given has EEG signals from subjects either (physically or imagining) moving their left or right hands. **We would like to construct a machine learning model that takes and EEG signal as an input and predicts whether the subject was trying to move their left or right hand** and the confidence behind that decision.

Our goal is to minimize the risk of our model associated with our dataset, while trying to have the model generalize well to data it hasn't seen before. We can define the risk as the average loss across the dataset. Picking a zero-one loss, where 1 is for an incorrect classification and 0 is for a correct classification[4], our **goal is then to maximize the accuracy of our model,** which is a one of many metrics to evaluate a classification model. This is defined as the ratio below[5].

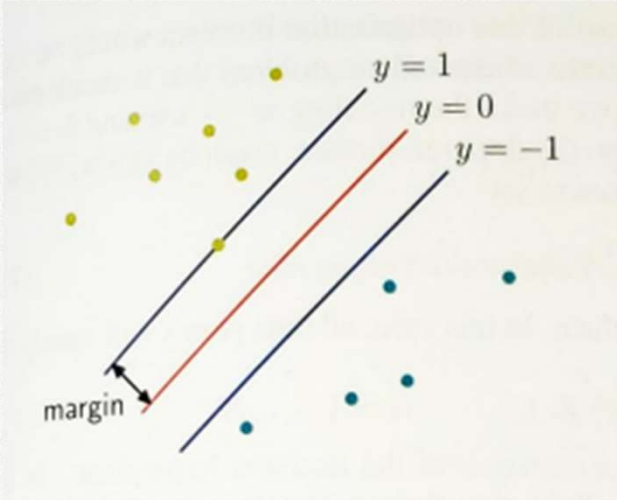
$$\text{Accuracy} = \frac{\text{Number of correct classifications}}{\text{Number of total datapoints}}$$

4. Claude, Sammut, and Webb I Geoffrey, editors. "Zero-One Loss." *Encyclopedia of Machine Learning*, 2011, pp. 1031–1031., https://doi.org/10.1007/978-0-387-30164-8_884.
5. (2011) Accuracy. In: Sammut C., Webb G.I. (eds) Encyclopedia of Machine Learning. Springer, Boston, MA. https://doi.org/10.1007/978-0-387-30164-8_3

Support Vector Machine (SVM) Basics

The support vector machine (SVM) is a model that takes multi-dimensional data and attempts to find a linear boundary between the two classes that maximizes the margin between the hyperplane boundary and the data. Below on the left is an example dataset of two classes (green and blue), with a linear boundary separating them[6].

On the right is the mathematical definition of the SVM, where phi is a transformation of the data, and y(x) is the distance between a transformed datapoint phi(x) and the boundary[6]. **w** and *b* are the parameters of the model, which need to be selected and optimized to get the maximum margin. **w** is a vector perpendicular to the boundary, so the inner product of it with a datapoint can represent the distance from the boundary.



$$y(x) = w^T \phi(x) + b$$

6. Bishop, Christopher M. "Sparse Kernel Machines." *Pattern Recognition and Machine Learning*, edited by Michael Jordan et al., Springer, 2006, pp. 325–338.

SVM Optimization

The margin is the perpendicular distance from the boundary to the closet point, x_n, from the dataset. **The goal of the SVM to optimize the parameters w and *b* such that the margin is maximized**. This optimization goal is shown to the right[6].

$$\arg\max_{\mathbf{w},b}\left\{\frac{1}{\|\mathbf{w}\|}\min_n\left[t_n\left(\mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x}_n)+b\right)\right]\right\}$$

The minimized term here is the closest distance from the boundary to a point, which we can define as 1, and then add the constraint that all points must be a distance of 1 or larger away from the boundary. This constraint is shown mathematically below on the right. The optimization problem then becomes minimizing the size of the vector **w**. Minimizing the size of the vector is the equivalent to minimizing the squared norm[6].

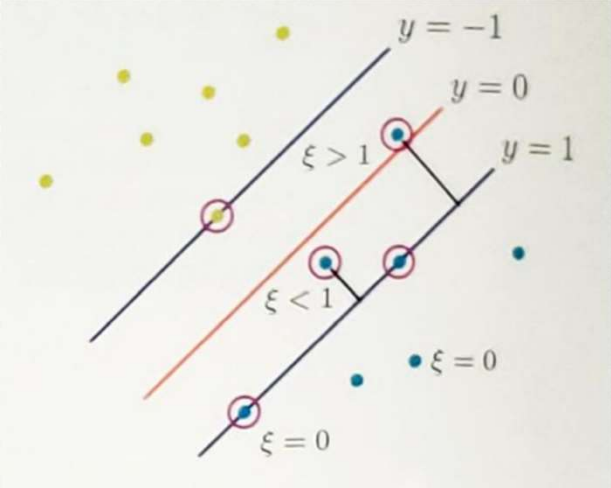
$$\text{argmin}_{w,b}\,\tfrac{1}{2}\|w\|^2 \text{ subject to } t_n(w^T\phi(x_n)+b)\leq 1, n=1,\ldots,N$$

This is a quadratic minimization problem subject to linear inequality constraints. This can be solved using Lagrange multipliers, which I will not go into, but is explained in the source below[6].

6. Bishop, Christopher M. "Sparse Kernel Machines." *Pattern Recognition and Machine Learning*, edited by Michael Jordan et al., Springer, 2006, pp. 325–338.

SVM Optimization for Non-Linearly Separable Data

However, if the data is not linearly separable, the optimization goal changes slightly. Slack variables, epsilon, introduce a penalty that increases linearly with distance from the boundary for datapoints on the wrong side of the boundary. **The new optimization goal is to maximize the margin while minimizing the penalty from points that are on the wrong side of the boundary**. This goal is described mathematically below on the right and shown in the diagram on the left[6]. The parameter C controls the weighting between the penalty from the slack variables and the margin, or in other words controls the amount of slack in the margin. In our experiments, **we will let lambda = 1/C, and try to optimize lambda by trying various values for this parameter and choosing the one with the best results**. For a small lambda, there will be high penalty for misclassifications and the boundary will try to have no classification errors, and for a large lambda, there will be little penalty for misclassification, so the model with have more bias rather than variance/complexity.



$$\text{argmin}_{w,b} C \sum_{n=1}^{N} \xi_n + \frac{1}{2}\|\mathbf{w}\|^2$$

$$\text{argmin}_{w,b} \frac{1}{\lambda} \sum_{n=1}^{N} \xi_n + \frac{1}{2}\|\mathbf{w}\|^2$$

6. Bishop, Christopher M. "Sparse Kernel Machines." *Pattern Recognition and Machine Learning*, edited by Michael Jordan et al., Springer, 2006, pp. 325–338.

Why Use an SVM?

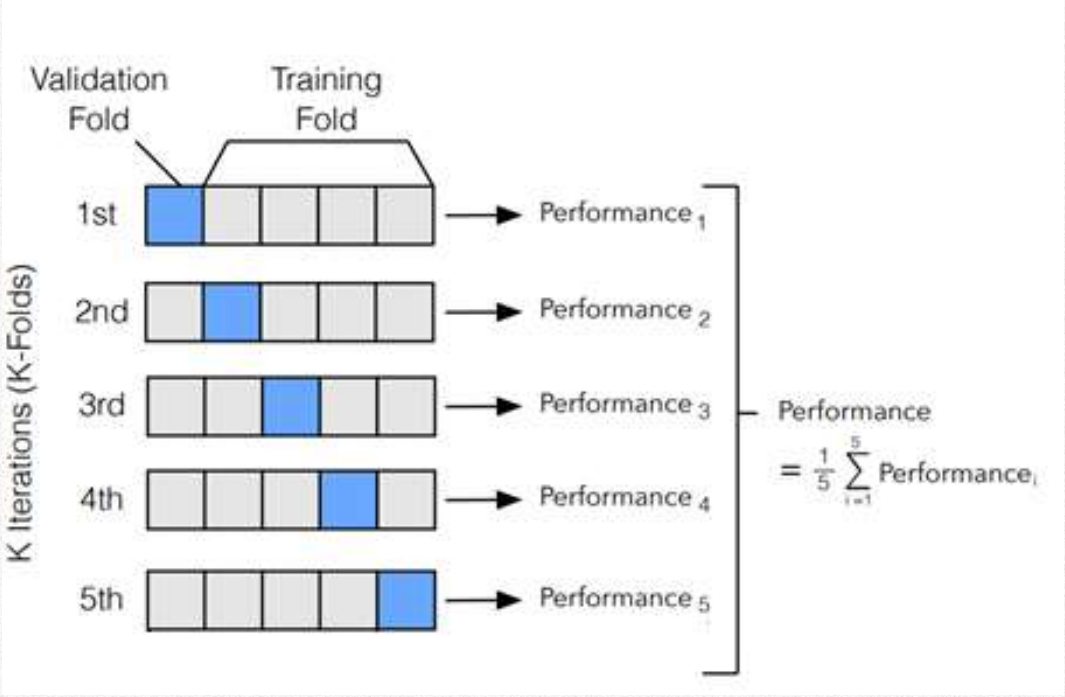
Our EEG dataset consists of 204 features, with 240 datapoints. Each feature is a dimension in the feature space. A typical rule of thumb in machine learning is that there should be at least 5 training examples for each dimension in the feature space[7]. Clearly our dataset has much fewer datapoints than what the heuristics say, meaning that **high dimensionality could be an issue with our model when trained on this dataset**. Specifically, high dimensionality increases the possible space of outcomes, causing the "ratio of an object's nearest neighbor over its farthest neighbor [to approach] one," which makes the data more easily separable than before, possibly leading to conclusions or results that won't generalize[8].

SVMs are effective in high-dimensional spaces, are memory efficient, and versatile[9]. **The effectiveness of the SVM for high-dimensional data is the primary advantage, since our dataset suffers from high-dimensionality**. This will mitigate the possibility of severely overfitting on the high dimensional data that may be separable just because of the number of dimensions in the data.

7. Theodoridis, Sergios, and Konstantinos Koutroumbas. *Pattern Recognition, 4th Edition*. Academic Press, 2009.
8. 4. Keogh E., Mueen A. (2017) Curse of Dimensionality. In: Sammut C., Webb G.I. (eds) Encyclopedia of Machine Learning and Data Mining. Springer, Boston, MA. https://doi.org/10.1007/978-1-4899-7687-1_192
9. Roy, Kunal, et al. "Selected Statistical Methods in QSAR." *Understanding the Basics of QSAR for Applications in Pharmaceutical Sciences and Risk Assessment*, 2015, pp. 191–229., https://doi.org/10.1016/b978-0-12-801505-6.00006-5.

Cross Validation

To evaluate the performance of the SVM on our dataset, we use cross validation. **Cross validation allows us to better understand and evaluate the accuracy of a model**. By using different training and testing sets, we can evaluate how much variance there is in model performance and get the average accuracy over various testing datasets. There are different methods of cross validation, but I chose to use K-fold cross validation. **K-fold cross validation involves splitting the data into *k* folds, which each take turn acting as the test set, with the other folds as the training data**. The *k* sets of results can then be averaged to get the performance of the model, as shown in the figure below[10].

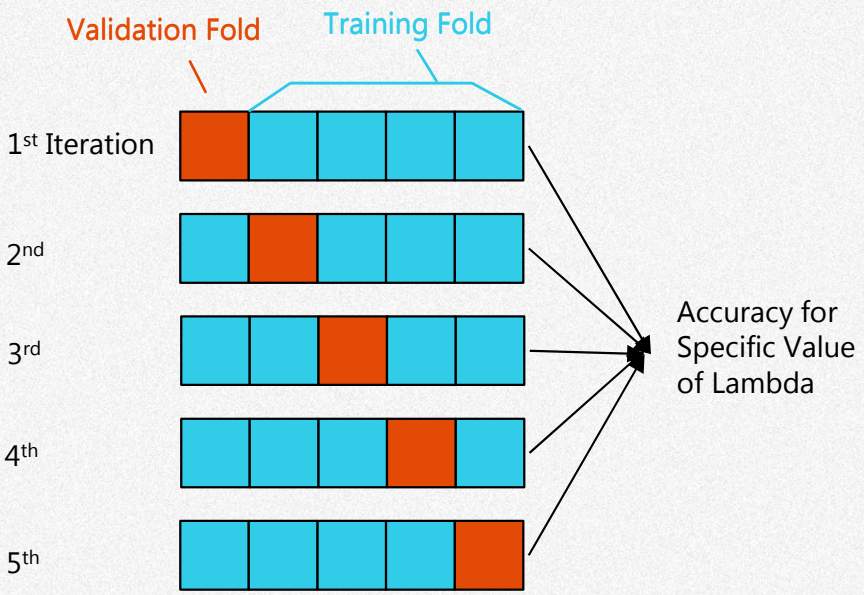


10. Raschka, Sebastian. "Model Evaluation, Model Selection, and Algorithm Selection in Machine Learning". 2018. 1811.12808.pdf (arxiv.org)

Hyperparameter Optimization via Cross Validation

The SVM has a regularization term that has a weight lambda, which is a hyperparameter that we can set. To determine the optimal value for lambda, we can iterate through a list of values for lambda and for each one, perform cross validation. We can then compare the accuracies for the different values for lambda and select the best one.

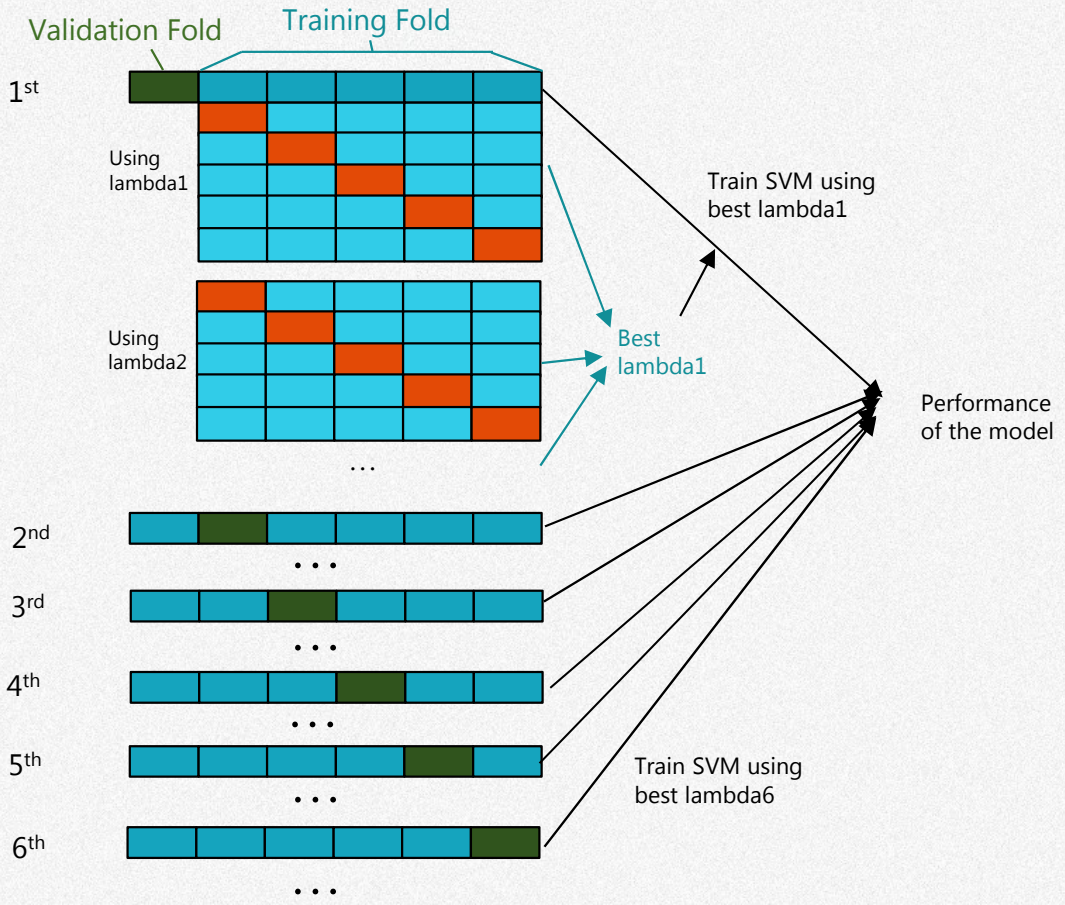
For each candidate value for lambda:




```
Data: dataset, lambdaList
lambdaAccuracies = []
for lambda ∈ lambdaList do
    crossValAccuracyList = []
    for trainingSet, testSet ∈ split(dataset) do
        crossValAccuracy = trainAndTestSVM(trainingSet, testSet,
         lambda)
        crossValAccuracyList.append(crossValAccuracy)
    end
    lambdaAccuracies.append(mean(crossValAccuracyList))
end
bestLambda = lambdaList[argmax(lambdaAccuracies)]
```

Two-Level Cross Validation

In this experiment, we will do two levels of cross validation. The first/outer level of cross validation will be to evaluate the performance of the model as described in slide 9, and the second/inner level will be to select the best value of lambda, as described in the previous slide, for each of the iterations in the first level of cross validation. The best value of lambda found by the inner level will be used during the training on the outer training fold. At the end of this process, we will have an overall performance of our chosen model.



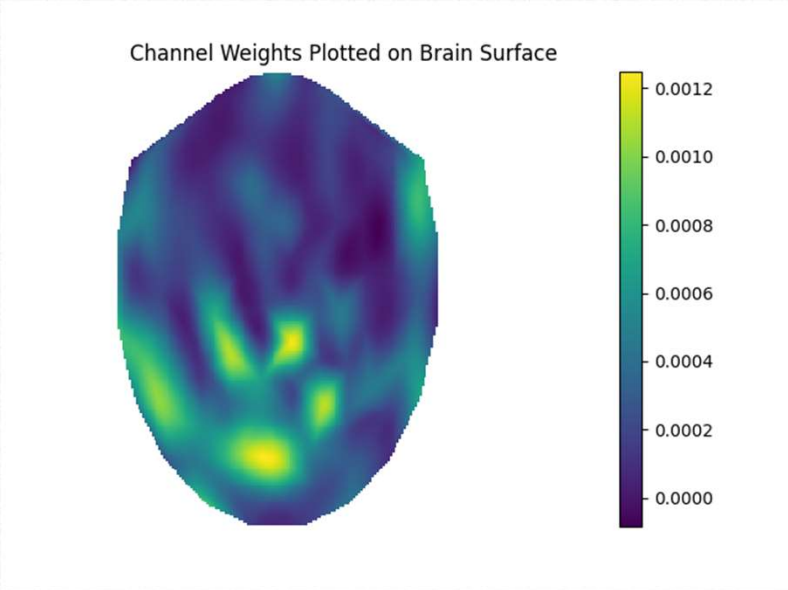

```
Data: dataset, lambdaList
crossVal1AccuracyList = []
for trainingSet1, testSet1 ∈ split(dataset) do
    lambdaAccuracies = []
    for lambda ∈ lambdaList do
        crossVal2AccuracyList = []
        for trainingSet2, testSet2 ∈ split(trainingSet1) do
            crossVal2Accuracy = trainAndTestSVM(trainingSet2,
              testSet2, lambda)
            crossVal2AccuracyList.append(crossVal2Accuracy)
        end
        lambdaAccuracies.append(mean(crossVal2AccuracyList))
    end
    bestLambda = lambdaList[argmax(lambdaAccuracies)]
    crossVal1Accuracy = trainAndTestSVM(trainingSet1, testSet1,
      bestLambda)
    crossVal1AccuracyList.append(crossVal1Accuracy)
end
accuracy = mean(crossVal1AccuracyList)
```

Experimental Results

Channel Weight

During training, the SVM learns weights associated with each feature of the data. Since each feature corresponds with an electrode/channel, **we can visualize the strength of these channels on the brain** based on the placement of the electrodes from the original experiment[11].

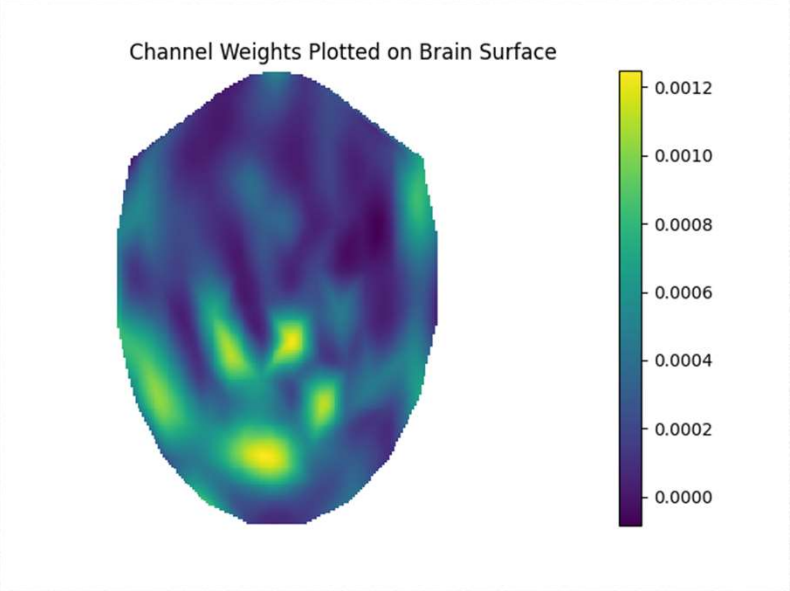
A large amplitude weight for a channel means that the channel is particularly useful for predicting what direction the subject would like to select. Since we have data from overt and imagined movement, we can see if different parts of the brain are activated when physically moving or simply imagining moving.

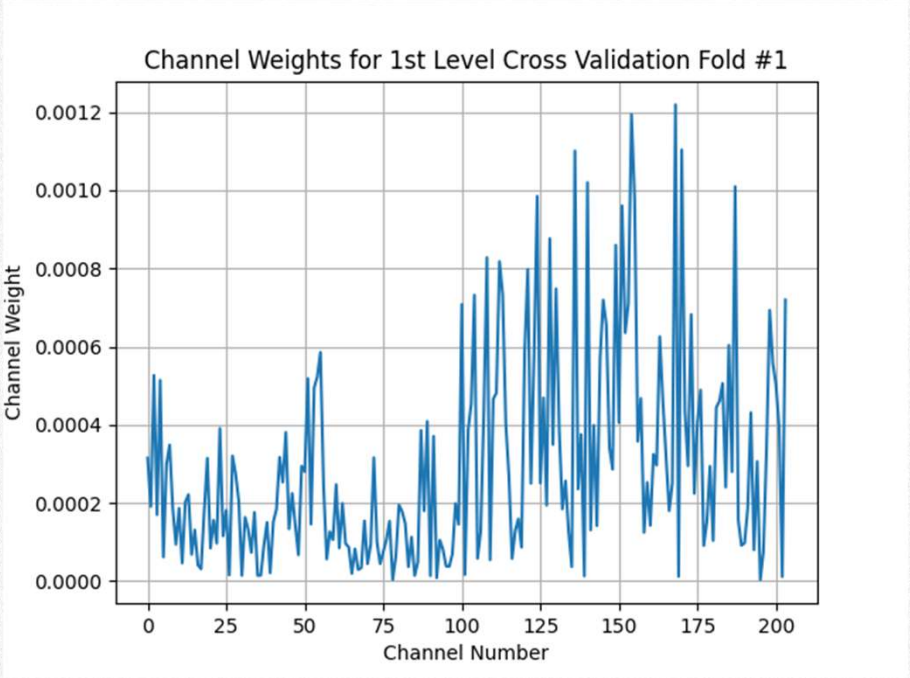


11. Tantum, Stacy. "show_chanWeights.py" Introduction to Machine Learning. Spring 2022. Accessed 24 April 2022.

Results for Overt Dataset

After training the SVM on the overt dataset, for the first cross-validation fold, we can visualize the strength of the SVM's channel weights. We see that the strong electrode channels are in the middle bottom of the diagram. We will see that the channel weights are relatively weak compared to those for the SVM trained on the imagined dataset.



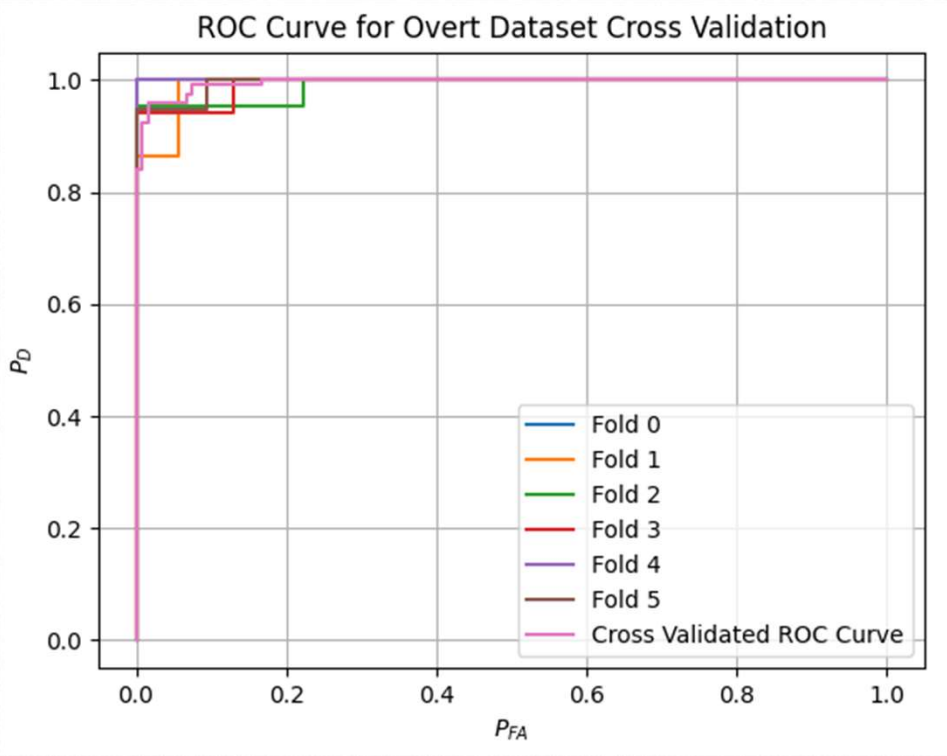


Channel Number	Channel Weight
140	0.00101
136	0.00110
170	0.00110
154	0.00119
168	0.00121

Results for Overt Dataset

Below you can see the ROC curve and accuracy for each outer cross validation fold, as well as the aggregated/average accuracy and ROC curve. The further the ROC curve reaches the upper left, the better the performance of the model. This shows that the performance on the overt dataset is exceptional.

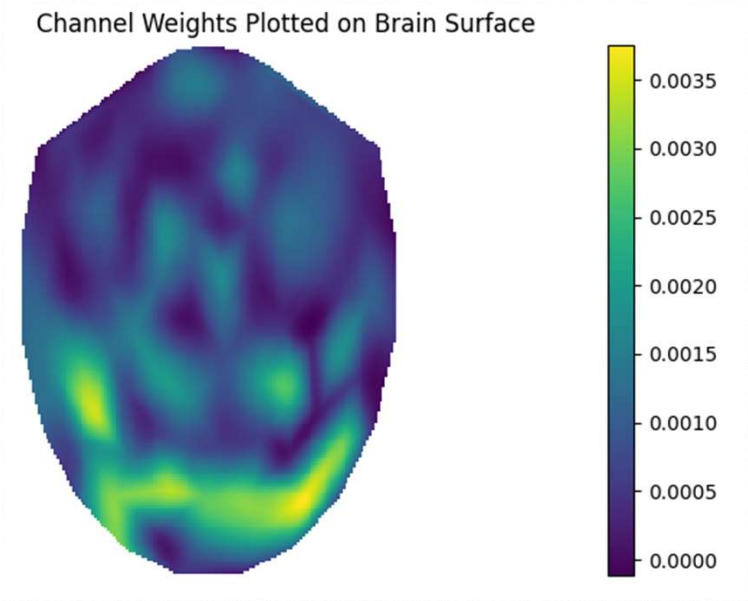
We can also see that the same value of lambda was picked. This value was also the smallest value of lambda that was tried. This corresponds with low regularization, allowing the model to be more complex.

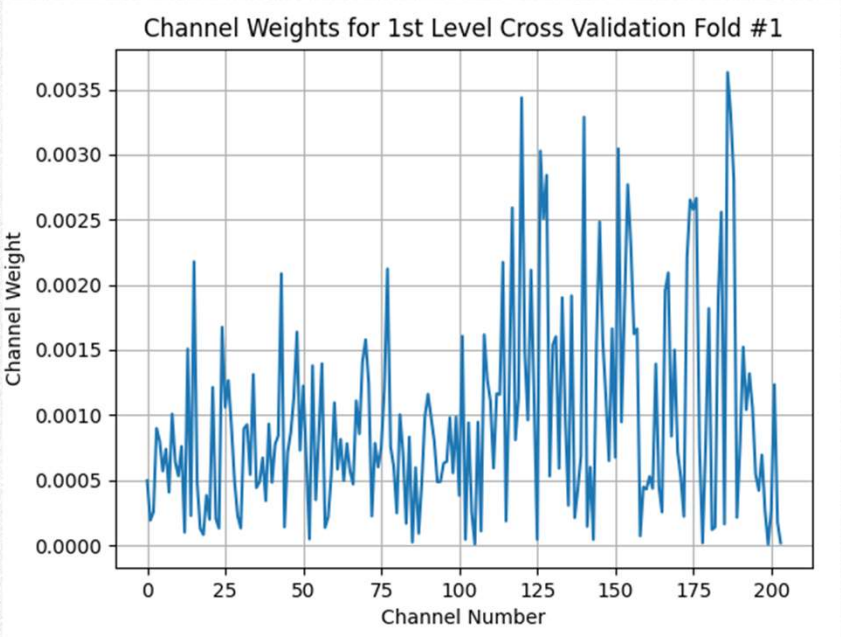


Fold	Accuracy	Best Value for Lambda
1	0.95	0.01
2	0.95	0.01
3	0.95	0.01
4	0.975	0.01
5	0.975	0.01
6	0.975	0.01
Average	0.954	N/A

Results for Imagined Dataset

After training the SVM on the overt dataset, for the first cross-validation fold, we can visualize the strength of the SVM's channel weights. As opposed to the strong channel weights closer to the middle of the brain for the overt dataset, we can see the strong channels are present in the semi-circular area in the bottom of the brain diagram. The channel weights here are much larger than those for the SVM trained on the overt dataset.



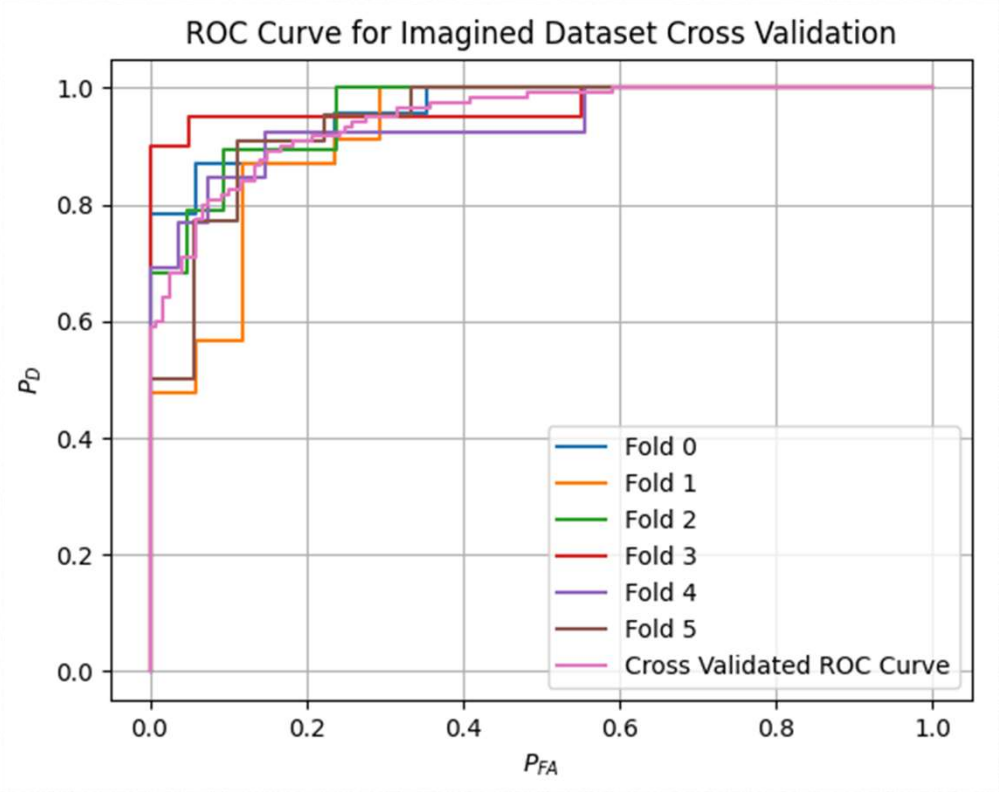


Channel Number	Channel Weights
151	0.00304
140	0.00328
187	0.00331
120	0.00343
186	0.00363

Results for Imagined Dataset

Below you can see the accuracy, ROC curves, and best values for lambda for each of the 6 folds for the 1st level of cross validation, as well as the averaged ROC and accuracy. We can see here that the ROC curve doesn't reach into the top left corner as far as the ROC curve for the SVM trained on the overt dataset.

Here the best selected value for lambda varied between folds, as opposed to the SVM trained on the overt dataset. The selected value of lambda here is either the largest or smallest possible value



Fold	Accuracy	Best Value for Lambda
1	0.825	0.01
2	0.825	0.01
3	0.85	0.01
4	0.95	0.01
5	0.875	0.01
6	0.85	0.01
Average	0.8625	N/A

Additional Experiments – Trying Different Kernels

The SVM allow the use of kernel functions, in which the input *x* is first mapped into a higher-dimensional feature space before the linear boundary is constructed[9]. This can introduce a non-linearity in the model, which can be useful if the data isn't linearly separable.

In this experiment, I tried four different kernels: linear, polynomial, radial basis function / gaussian, and sigmoid. Each of these kernels were used 10 times for the full 2-level cross validation, and then averaged. I found that **using any non-linear kernel worsens performance relative to a linear kernel**. This is likely because the data already has so many features that mapping it to a higher-dimensional feature space introduces additional complexity to the model that can cause lack of generalization.

Kernel	Overt Dataset Accuracy	Imagined Dataset Accuracy
Linear	0.957	0.880
Polynomial	0.891	0.810
RBF	0.932	0.858
Sigmoid	0.919	0.804

9. Roy, Kunal, et al. "Selected Statistical Methods in QSAR." *Understanding the Basics of QSAR for Applications in Pharmaceutical Sciences and Risk Assessment*, 2015, pp. 191–229., https://doi.org/10.1016/b978-0-12-801505-6.00006-5.

Additional Experiments – Subset of Features

To reduce the dimensionality of the dataset, I tried feature selection. After running cross validation, I found the strongest channels and selected a varying number of them to see how the performance would be affected. As the number of channels is decreased, less information is available to the model, but the issues of high dimensionality are reduced, so there is a trade-off.

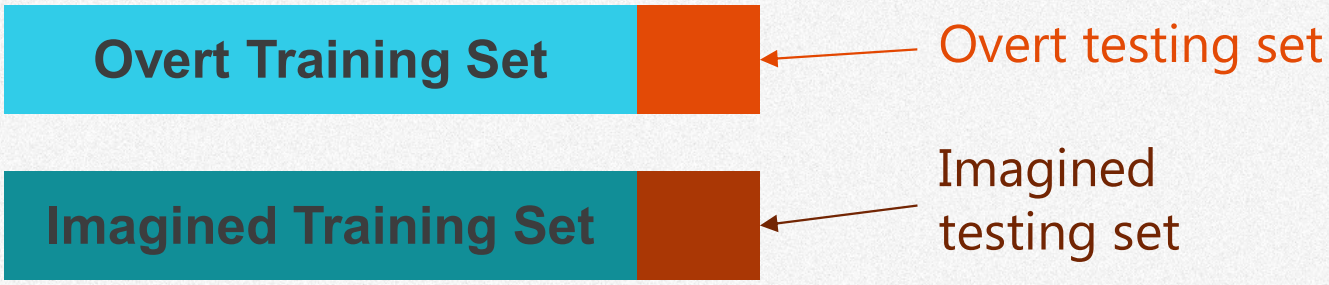
From the table of results below, I found that **reducing the dimensionality had significant benefits to the performance of the model**. When using 60 and 90 of the strongest channels, the results for both the overt and imagined dataset significantly improved. For the overt dataset, they improved by about 3%, and about 5.5% for the imagined dataset.

Because of the marked improvement in performance, we can see that **most of the EEG channels do not carry significantly useful information, so removing them helps with the dimensionality more than hurting by removing information**. We do see that with 30 channels left, too much information is removed, and the performance starts dropping.

Number of Channels	Accuracy for Overt Dataset	Accuracy for Imagined Dataset
204	0.958	0.865
180	0.959	0.874
150	0.958	0.894
120	0.975	0.894
90	0.982	**0.919**
60	**0.985**	0.910
30	0.949	0.883

Additional Experiments – Training and Testing on Different Datasets

To experiment with the differences in the two datasets, I created an experiment to train on one dataset while testing on the other. The two datasets were both split randomly into a training and testing set. I then trained an SVM with a single level of cross validation to select the best value of lambda, and then tested on one of these datasets. The results of the four possible combinations are shown in the table below. We can see that **when testing on imagined, the SVM performs poorly, but when testing on overt, even when having trained on the imagined dataset, the SVM performs well**.



	Test Overt	Test Imagined
Train Overt	0.946	0.804
Train Imagined	0.898	0.897

Discussion and Conclusions

Results from Overt vs. Imagined Dataset

The performance of the SVM on the overt dataset is much better than that on the imagined dataset. The accuracy of the SVM for the overt dataset is about 10% higher than that for the imagined dataset. This intuitively makes sense, since physically moving a hand would require more control from the brain than imagining it moving. This is consistent with previous research studies where it has been found that the task of classification of motion is easier for physical than imagined[1]. This "can be justified by the fact that actual motion performance is manifested in more consistent manner in the brain activity, compared to a strictly mental activity of imagining the motion without any helpful feedback (visual or sensory)"[12]. This means that the EEG signals will be more easily separable for the overt dataset than the imagined.

The average number of support vectors for the overt dataset SVM was 34, while for the imagined dataset SVM, it was 45. This is likely because the imagined dataset is more complex, and not as easily separable as the overt dataset, as discussed earlier. **If the imagined dataset is more complex, then there will likely be more datapoints close to the boundary, then there will be more support vectors since they will lie within the margin**. Both SVMs had the smallest value for lambda, so that isn't a factor to consider, just the distribution of the data.

We also saw that the weights for the SVM for the imagined dataset are much larger, likely due to the weaker nature of the imagined EEG signals.

Dataset	Accuracy
Overt	0.954
Imagined	0.863

12. Szczuko, Piotr. "Real and Imaginary Motion Classification Based on Rough Set Analysis of EEG Signals for Multimedia Applications." *Multimedia Tools and Applications*, vol. 76, no. 24, 2017, pp. 25697–25711., https://doi.org/10.1007/s11042-017-4458-7.

Factors That Impact Classification Accuracy

Overt vs. imagined movement – We saw from the experiments that the imagined dataset is much more difficult for the model to learn and to test on. When training and evaluating with the overt dataset, the accuracy was about 10% larger than that for the imagined dataset. We saw from the experiment training and testing on different datasets that the imagined movement data could not be predicted well by either the SVM trained on the imagined data or the overt data.

Dimensionality of the dataset – From the experiment with the number of channels to use, we can see that changing the dimensionality of the data can have significant results on the classification accuracy. Decreasing the number of EEG channels to only use those that are strongest improved the accuracy significantly until about 30 channels. Using 60 or 90 channels resulted in the best performance, particularly for the imagined data. This indicates that the imagined data might be complex within the high dimensional space, resulting in issues generalizing to test data. The benefit of reducing the dimensionality of the data outweighed the loss of information in the data, up to about 30 channels.

Kernel chosen for SVM – From the additional experiment with various kernels, we can see that adding a nonlinear kernel to the SVM worsened performance significantly. This is likely because of increasing the dimensionality of the data through the kernel. We have seen that reducing the dimensionality of the data helps, so increasing the dimensionality of the data through the kernel intuitively will worsen performance.

Limits With The Approach

Performance with Imagined Data – From our experiments, we saw that the approach fails to perform particularly well on the data where the subjects are imagining moving their left/right hands instead of physically moving them. This might be unavoidable in the case that the EEG signals from a subject imagining moving their hand aren't definitive or conclusive enough for extremely accurate classification. However, it might be the case that the linear boundary constraint of the SVM might introduce bias to the model, resulting in systematic error.

Lack of Domain Knowledge Utilized – This problem we are tackling is deeply within the realm of neuroscience. There has been vast amounts of research published on EEG signals and applying machine learning techniques to EEG signals[13] and failing to incorporate this research might result in sub-optimal performance for our machine learning model. Analyzing research on EEG signals for physical vs. imagined movements, and EEG signals for left vs. right movements could lead to useful ideas to incorporate into our machine learning model, feature selection process, or a pre-processing step for our EEG signals.

13. Hosseini, Mohammad-Parsa, et al. "A Review on Machine Learning for EEG Signal Processing in Bioengineering." *IEEE Reviews in Biomedical Engineering*, vol. 14, 2021, pp. 204–218., https://doi.org/10.1109/rbme.2020.2969915.

Improvements Made

Using a subset of the features – By using a subset of the features, the performance of the SVM model trained and evaluated on the overt data improved by 0.027, or 2.82%. The performance of the SVM model trained and evaluated on the imagined dataset improved by 0.054, or an increase of 6.24%. Below are the raw improvements to accuracy and percentage increase from the baseline of 204 channels. We can see that reducing the dimensionality of the data much outweighs the loss of information from removing EEG channels, likely because these removed channels don't carry important information to help with the classification.

Number of Channels	Accuracy for Overt Dataset	Improvement	Accuracy for Imagined Dataset	Improvement
204	0.958	+0.00 (0%)	0.865	+0.0 (0%)
180	0.959	+0.001 (0.001%)	0.874	+0.009 (1.04%)
150	0.958	+0.0 (0%)	0.894	+0.029 (3.35%)
120	0.975	+0.017 (1.77%)	0.894	+0.029 (3.35%)
90	0.982	+0.024 (2.5%)	**0.919**	**+0.054 (6.24%)**
60	**0.985**	**+0.027 (2.82%)**	0.910	+0.045 (5.20%)
30	0.949	-0.009 (0.94%)	0.883	+0.018 (2.08%)

Possible Further Improvements

Utilizing Domain Knowledge – Knowing more about the neuroscience behind the observed data could likely provide intuition to improve our model. For example, if we knew what lobes or areas of the brain are used for left/right hand movement or for physical/imagined movement, then that could be incorporated into the model. We can look at the strength of the EEG channels to infer this information, but it may be helpful to include this information in the model so that the ML model doesn't need to learn this during training.

Feature Extraction – Extracting features out of the dataset, such as the mean and standard deviation possibly within each lobe, or other pertinent information and replacing some of the raw channel features could improve performance. It would reduce dimensionality by removing some of the raw channels in the data while hopefully adding useful information to make a classification. This might require some domain knowledge to know what summary information would be useful for differentiating left vs. right hand movement. Although this would be for time series EEG signals, an example feature extraction technique is using wavelet transformations[14].

14. Amin, Hafeez Ullah, et al. "Classification of EEG Signals Based on Pattern Recognition Approach." *Frontiers in Computational Neuroscience*, vol. 11, 2017, https://doi.org/10.3389/fncom.2017.00103.

Possible Further Improvements

Incorporating Time-Series Data – Although this isn't within the realm of opportunity within our dataset, for a more advanced BCI it would make sense to do classifications on time series data. This is because EEG signals will vary throughout time, so it would be more realistic and useful to have a machine learning model that can work on time series data. Previous research has shown that this time series data can aid in complicated tasks like the diagnosing patients[15], and so it can likely be useful for left/right movement classification.

Trying Other Models After Feature Extraction / Dimensionality Reduction – We chose the SVM particularly because of its effectiveness with high-dimensional data. However, we found that only a small subset (around 1/4) of the channels carry significantly useful information for our task. After reducing the dimensionality of our dataset, we could try alternative models, such as Bayes classifiers, relevant vector machines, etc…, which may not have been able to perform well with the previously high dimensional data. This could help model the imagined data, since this data may be too complex for the linear boundary of the SVM to perform well.

15. Dutta, Kusumika Krori. "Multi-Class Time Series Classification of EEG Signals with Recurrent Neural Networks." *2019 9th International Conference on Cloud Computing, Data Science & Engineering (Confluence)*, 2019, https://doi.org/10.1109/confluence.2019.8776889.

References

References

1. Kumar, J. Satheesh, and P. Bhuvaneswari. "Analysis of Electroencephalography (EEG) Signals and Its Categorization–A Study." Procedia Engineering, vol. 38, 2012, pp. 2525–2536., https://doi.org/10.1016/j.proeng.2012.06.298.

2. Kline, Adrienne, et al. "EEG Differentiates Left and Right Imagined Lower Limb Movement." Gait & Posture, vol. 84, Feb. 2021, pp. 148–154., https://doi.org/10.1016/j.gaitpost.2020.11.014.

3. Saha, Simanto, et al. "Progress in Brain Computer Interface: Challenges and Opportunities." Frontiers in Systems Neuroscience, vol. 15, 25 Feb. 2021, https://doi.org/10.3389/fnsys.2021.578875.

4. Claude, Sammut, and Webb I Geoffrey, editors. "Zero-One Loss." Encyclopedia of Machine Learning, 2011, pp. 1031–1031., https://doi.org/10.1007/978-0-387-30164-8_884.

5. (2011) Accuracy. In: Sammut C., Webb G.I. (eds) Encyclopedia of Machine Learning. Springer, Boston, MA. https://doi.org/10.1007/978-0-387-30164-8_3

6. Bishop, Christopher M. "Sparse Kernel Machines." Pattern Recognition and Machine Learning, edited by Michael Jordan et al., Springer, 2006, pp. 325–338.

7. Theodoridis, Sergios, and Konstantinos Koutroumbas. Pattern Recognition, 4th Edition. Academic Press, 2009.

8. Keogh E., Mueen A. (2017) Curse of Dimensionality. In: Sammut C., Webb G.I. (eds) Encyclopedia of Machine Learning and Data Mining. Springer, Boston, MA. https://doi.org/10.1007/978-1-4899-7687-1_192

References

9. Roy, Kunal, et al. "Selected Statistical Methods in QSAR." Understanding the Basics of QSAR for Applications in Pharmaceutical Sciences and Risk Assessment, 2015, pp. 191–229., https://doi.org/10.1016/b978-0-12-801505-6.00006-5.

10. Raschka, Sebastian. "Model Evaluation, Model Selection, and Algorithm Selection in Machine Learning". 2018. 1811.12808.pdf (arxiv.org)

11. Tantum, Stacy. "show_chanWeights.py" Introduction to Machine Learning. Spring 2022. Accessed 24 April 2022.

12. Szczuko, Piotr. "Real and Imaginary Motion Classification Based on Rough Set Analysis of EEG Signals for Multimedia Applications." Multimedia Tools and Applications, vol. 76, no. 24, 2017, pp. 25697–25711., https://doi.org/10.1007/s11042-017-4458-7.

13. Hosseini, Mohammad-Parsa, et al. "A Review on Machine Learning for EEG Signal Processing in Bioengineering." IEEE Reviews in Biomedical Engineering, vol. 14, 2021, pp. 204–218., https://doi.org/10.1109/rbme.2020.2969915.

14. Amin, Hafeez Ullah, et al. "Classification of EEG Signals Based on Pattern Recognition Approach." Frontiers in Computational Neuroscience, vol. 11, 2017, https://doi.org/10.3389/fncom.2017.00103.

15. Dutta, Kusumika Krori. "Multi-Class Time Series Classification of EEG Signals with Recurrent Neural Networks." 2019 9th International Conference on Cloud Computing, Data Science & Engineering (Confluence), 2019, https://doi.org/10.1109/confluence.2019.8776889.

References – Python Packages

"Matplotlib.pyplot." Matplotlib.pyplot - Matplotlib 3.5.0 Documentation, https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.html. Accessed 23 April 2022.

NumPy, https://numpy.org/. Accessed 23 April 2022.

"Sklearn.svm.LinearSVC." Scikit, https://scikit-learn.org/stable/modules/generated/sklearn.svm.LinearSVC.html#sklearn.svm.LinearSVC. Accessed 23 April 2022.

"Sklearn.model_selection.Kfold." Scikit, https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.KFold.html. Accessed 23 April 2022.

Tantum, Stacy. "show_chanWeights.py" Introduction to Machine Learning. Spring 2022. Accessed 24 April 2022.

Collaborations

Collaborations

I collaborated with the project group I was assigned to, which had Kevin Tu, Eric Qi, and Shaan Gondalia.

I shared and debated ideas with: Kevin, Eric, and Shaan. During our peer feedback sessions, we discussed ideas for next steps each time and possible extension experiments to try. I talked especially with Eric about what extension experiments to try, and what results we should expect from those and the reasoning for that.

I did not share or receive code during this project. I did discuss packages to use for cross validation and the general structure of the program with my group members during the peer feedback sessions.

I compared results with: Kevin, Eric, and Shaan during the peer feedback sessions to make sure that we were all getting results consistent with one another. For the extension experiments, I compared results with Eric to make sure that we were both doing these experiments correctly.

Who did you help overcome an obstacle: I helped Kevin with questions about why SVMs are particularly useful for this dataset and why we would choose to use them.

Who helped me overcome obstacles: Eric helped me by verifying results I got, which allowed me to catch when something in my experiments was not working.