Brain Computer Interface Movement Decoding

Arnav Nayak
ECE 580: Introduction to Machine Learning

Dr. Stacy Tantum

Overview

Project Description

Background

Brain Computer Interfaces - Why it Matters?

Brain computer interfaces (BCIs) are computer-based systems that acquire brain signals, analyze them and translate them into commands in order to carry out desired actions. BCIs establish a direct communication pathway between the human brain and external devices, allowing control of these devices through brain signals alone. As we progress further into a technology-driven era, BCIs offer a path forward to **redefine human computer interaction** and **enhance our relationship with the world around us**. For example, BCIs are particularly transformative for individuals with neuromuscular disabilities because they enable control over assistive technologies such as prosthetic limbs, computer cursors, and communication aids, restoring useful function and unlocking a better quality of life.[1] Extending beyond medical uses, BCIs hold tremendous potential in a variety of mainstream applications, facilitating new ways of machine interaction such as allowing users to control everyday technology with only thoughts. Overall, BCI technology has the potential to **expand the limits of human capability**, transforming our interaction with digital environment.

[1]Shih, Jerry J et al. "Brain-computer interfaces in medicine." Mayo Clinic proceedings vol. 87,3 (2012): 268-79. doi:10.1016/j.mayocp.2011.12.008

Background

Curse of Dimensionality

We might think that, almost intuitively, more information is always better for any data-based predictive model. In the 'big data' era, there is no shortage of **high dimensional data**. However, the sheer number of variables, or features, that is collected from a single sample can be problematic. In machine learning, the curse of dimensionality refers to the various challenges that occur when we **increase the number of features in a dataset without increasing the number of observations** to characterize those features. This high dimensionality can dilute the effectiveness of traditional machine learning algorithms due to the **sparsity of the data** in such a large feature space. When there are a lot of features and relatively few observations, it is easy for a model to find spurious relationships in the data that do not exist, resulting in overfitting.[2]

This phenomenon is particularly relevant when dealing with EEG data in the context of brain-computer interfaces). EEG datasets are inherently high-dimensional, with multiple electrodes capturing brain activity simultaneously, resulting in a large number of features for each observation. For instance, in this project, a single observation is represented by a **204-dimensional vector**, but we only have 120 observations per dataset. Since the number of features outnumbers the number of observations, we must account for the curse of dimensionality in our classification approach.

[2]Altman, N., Krzywinski, M. The curse(s) of dimensionality. Nat Methods 15, 399–400 (2018). https://doi.org/10.1038/s41592-018-0019-x

Background

Support Vector Machines

A support vector machine (SVM) is a machine learning algorithm that uses supervised learning models to solve a binary classification task. The fundamental idea behind SVM is to find the **hyperplane that best separates different classes** in the dataset. An SVM model represents the examples as points in space, and the optimal hyperplane is calculated so that the examples of separate categories are divided by a clear gap that is as wide as possible. New examples are then predicted to belong to a category based on which side of the hyperplane they fall on. In essence, SVMs attempt to find the **"maximum margin"** hyperplane—the one that makes the largest separation between classes. In order to calculate this hyperplane, SVM relies on a subset of the training observations, referred to as the **support vectors**.[3]

Why SVMs can be effective with high dimensional data even when relatively few training observations are available?

We know that the **curse of dimensionality** can cause a model to overfit to datasets with high dimensionality. However, SVMs are particularly well-suited for high-dimensional data spaces, like those presented by EEG datasets in BCI applications, because of the following traits:

- **Margin Maximization**: In high-dimensional spaces, even if the data points are not linearly separable in lower dimensions, it's more likely that they can be separated linearly. Thus, margin maximization is effective in higher dimensions.
- **Sparsity of Solution**: Since SVM relies on only a subset of training observations, SVM does not necessarily become more complex as the number of dimensions increases. Rather, its complexity is determined by the number of support vectors.
- **Regularization**: SVMs inherently include regularization parameters which helps in avoiding overfitting, a common problem in high-dimensional datasets.

[3]Hsu, Chih-wei & Chang, Chih-chung & Lin, Chih-Jen. (2003). A Practical Guide to Support Vector Classification Chih-Wei Hsu, Chih-Chung Chang, and Chih-Jen Lin.

Project Goals

BCI Movement Decoding - Objectives

The goal of this project is to delve into BCI movement decoding by creating a predictive model that can determine whether a BCI user wishes to select class “left” or “right”. Specifically, we want to

- Create a binary classifier that can take in as input an **EEG observation** and determine if the observation aligns with classes **“left”** or **“right”**.
- Use **supervised learning** to train an SVM classifier with a **linear kernel.**
- Use **L1-Regularization (Lasso)** to enforce a sparsity constraint on the weights of the SVM classifier.
- Optimize the selection of the regularization parameter α through **Two-Level Cross Validation.**
- Test the SVM classifier under four distinct **training/testing scenarios** to assess its robustness and versatility.
- Visualize and interpret the resulting SVM and its corresponding **Support Vectors.**
- Visualize the **ROC curves** of each Cross Validation fold and compare the accuracy of different simulation scenarios.
- Assess the impact of using the **RBF** and **Polynomial** kernels for SVM.

Project Goals

BCI Movement Decoding - Project Data

For this project, we will be using **EEG signals** collected from an able-bodied human subject. The subject has 102 electrodes and their positions on the human head are shown to the right (*Figure 1*). Each electrode provides 2 quantities relevant to the gradient of the E-field in each direction. This means that each input observation is a **204-dimensional vector.**

We will be using two types of datasets: **imagined** movement and **overt** movement. The imagined dataset contains observations when the subject imagined moving their left or right hands, whereas the overt dataset contains observations when the subject actually moved their left or right hands. For each type of dataset, there are 120 observations for each class, for a total of 240 observations per dataset. The mapping of class to "left"/"right" is unknown, so we will refer to the classes as *H0* and *H1*. We will run 4 different simulations with a linear SVM, training and testing on each type of datasets (**same-train** & **cross-train**).

Using **both experimental conditions** are relevant to movement classification with BCIs because overt data provides clear EEG responses associated with actual movement, serving as ground truth to **validate BCIs**, while imagined data captures a user's intent, which simulates a real observation that a BCI has to classify.

We will be using Python to create our models. We will be using *scikit-learn*[4] to assist our implementations of the SVM classifier and Cross Validation. We will also make use of the tools provided to us by *numpy*[5] and *matplotlib*[6] libraries for computation and visualization.

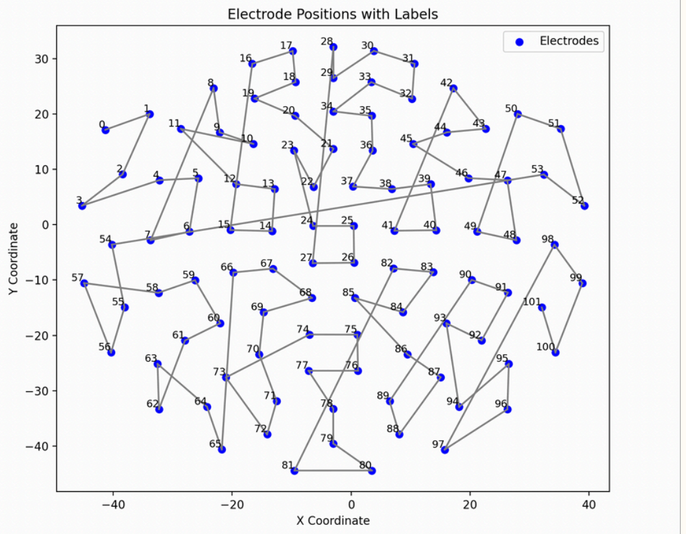


Figure 1: Electrode Positions

[4]Scikit-learn: Machine Learning in Python, Pedregosa et al., JMLR 12, pp. 2825-2830, 2011.
[5]Harris, C.R., Millman, K.J., van der Walt, S.J. et al. Array programming with NumPy. Nature 585, 357–362 (2020). DOI: 10.1038/s41586-020-2649-2.
[6]J. D. Hunter, "Matplotlib: A 2D Graphics Environment", Computing in Science & Engineering, vol. 9, no. 3, pp. 90-95, 2007.=

Project Goals

Note - Other Applications

Support Vector Machines can be used for a vast number of applications other than EEG movement decoding. One specific scenario is **text categorization**. Text categorization is the process of sorting text documents into one or more predefined classes of similar documents. Examples of text categorization tasks include **spam filtering** or **sentiment analysis.** In natural language processing, textual data is often represented with **high dimensional vectors**. If we can build a classifier that can handle high dimensional input, we can perform class predictions using textual data as input.[7]

Word embeddings can be used to convert words into vectors that convey semantic meaning. These word embeddings are usually very high dimensional in order to capture a wide range of **semantic relationships** and **nuances in language**. SVMs can be used with these word embeddings to perform classification tasks. For example, we can use the same linear kernel SVM approach to create a classifier that distinguishes between different types of **sentiments in customer reviews**, categorizing them as positive, negative, or neutral based on the semantic content of the text.[7]

[7]A. Basu, C. Walters and M. Shepherd, "Support vector machines for text categorization," 36th Annual Hawaii International Conference on System Sciences, 2003. Proceedings of the, Big Island, HI, USA, 2003, pp. 7 pp.-, doi: 10.1109/HICSS.2003.1174243.

Mathematical Formulation

Support Vector Machines

Support Vector Machines - What is it?

Support Vector Machines[7] are a type of **supervised machine learning** algorithm that can be used for both classification and regression tasks. As described previously, we chose SVMs because of their ability to handle **high dimensional data**. The main idea behind SVMs is that we want to construct a boundary that segregates the data points of different classes. In 2 dimensions, the boundary is a 1D line. In 3 dimensions, the boundary is a 2D plane. In *p* dimensions, we refer to the boundary as a *p-1* dimensional **hyperplane**.

For a given dataset, there are a number of hyperplanes that can successfully separate the observations. Intuitively, SVM defines the optimal hyperplane as the one that **maximizes the distance between the hyperplane and nearby training points**, known as the **margin**. A larger margin combats against overfitting and ensures that slight deviations in the data points do not affect the model. The figure on the right (*Figure 2*) shows the optimal hyperplane that maximizes the margin between two classes. The observations on the margin boundaries are called **support vectors.**

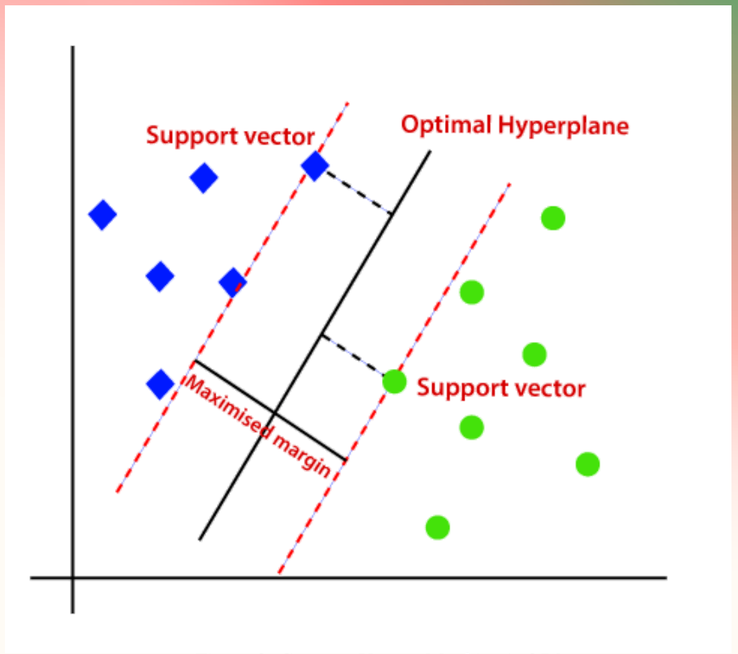


Figure 2: Support Vector Machine in 2D[8]

[7]Cortes, C., & Vapnik, V. (1995). Support-vector networks. Machine Learning, 20(3), 273–297.
[8]Salunkhe, Vivek. "Support Vector Machine (SVM)." Medium, Medium, 23 July 2021, medium.com/@viveksalunkhe80/support-vector-machine-svm-88f360ff5f38.

Support Vector Machines

Support Vector Machines - How do they work?

The equation of a hyperplane is given on the right (*Figure 3*), where ***W*** represents the coefficients of the hyperplane, ***X*** represents the features and ***C*** represents a bias term.

$$\left(W^T X_i + C\right)$$

Figure 3: Equation of Hyperplane[9]

The class determination of a new observation simply checks which side of the hyperplane the observations falls in. If $\mathbf{(W^T X_i + C) \geq 0}$, then the observation is on or above the hyperplane and the predicted class is +1. Otherwise, the observation is below the hyperplane and the predicted class is -1. In order to find the optimal hyperplane, SVM optimizes the following **objective function** (*Figure 4*).

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

$$\text{S.T. } y_i \cdot \left(W^T X_i + C\right) \geq 1 - \xi_i$$
$$\xi_i \geq 0$$
$$\left(i = 1, 2, \cdots, N\right)$$

Figure 4: SVM Objective Function[9]

The first term, $\boldsymbol{\xi i}$, is a **slack variable** associated with every data point that does not lie on the correct side of the margin. The slack variable quantifies how much the data point **violates the margin requirement**. Ideally, our data is perfectly separable by a hyperplane. In practice, however, this is rarely the case and outliers in data can exist. When SVM does not allow for any misclassification while calculating the optimal hyperplane, it is referred to as a hard margin SVM. In order to combat overfitting to outliers in the training data, we allow some misclassifications to occur. This is referred to as a **soft margin SVM**. The first term above minimizes the error caused by slack variables.

We subject the objective function to the **constraints** shown above. For each data point, the product of the true class and the class predicted by the decision function (which is always a positive value if the prediction is correct) should be greater than the margin, allowing for some misclassification to occur. Thus, the first term allows SVM to **minimize classification error.**

[9]Tantum, Stacy. “Brain Computer Interface Movement Decoding Lecture Slides” Duke University. Apr 2024.

Ridge (L2-Norm Regularization)

Ridge - What is it?

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

$$\text{S.T. } y_i \cdot \left(W^T X_i + C\right) \geq 1 - \xi_i$$
$$\xi_i \geq 0$$
$$\left(i = 1, 2, \cdots, N\right)$$

Figure 5: SVM Objective Function[11]

The second term in the objective function maximizes the margin by minimizing **W^TW.** This is the sum of the squares of the components of W, referred to as the L2 norm. Minimizing this term does 2 things. Firstly, the margin is inversely proportional to **$\|W\|$**, so by minimizing, **$\|W\|^2$**, the SVM is **maximizing the distance** between the hyperplane and the closest points from both classes, which are the support vectors. This results in a larger margin, which is our objective for SVM.

Secondly, minimizing the coefficient vector **W** also attempts to **minimize the value of the weights** that determine the hyperplane. This is a form of regularization called **Ridge (L2-Norm Regularization)[10]** that tends to produce sparse solutions, solutions with fewer non-zero coefficients. This reduces the number of features that have a significant impact on the classification decision, which can be beneficial in high-dimensional spaces to combat overfitting. The strength of regularization is controlled by the hyperparameter **α**.

Since this term acts primarily to maximize the margin, SVM inherently contains L2-Norm regularization. Thus, soft margin SVM finds a balance between **maximizing the margin**, **minimizing weight values**, and **minimizing classification error** through the use of slack variables.

[10]Hoerl, Arthur E., and Robert W. Kennard. "Ridge Regression: Biased Estimation for Nonorthogonal Problems." Technometrics, vol. 12, no. 1, 1970, pp. 55–67. JSTOR, https://doi.org/10.2307/1267351. Accessed 28 Apr. 2024.
[11]Tantum, Stacy. "Brain Computer Interface Movement Decoding Lecture Slides" Duke University. Apr 2024.

Movement Decoding

Two Level Cross Validation - Selecting Regularization Strength

The strength of the LASSO regularization depends on the **hyperparameter α**. This hyperparameter controls the trade off between the goodness of fit and the degree of regularization. Instead of using a single split to find our **regularization strength**, we can find the optimal regularization strength through **two level cross validation.**

In the first level of Cross Validation, we split our data into 6 **stratified folds** and hold each fold out once for testing. On each iteration, we use a second level of 5-fold Cross Validation to determine the optimal value of **α** for that fold, using accuracy as our performance metric. In the end, we will find 6 values of **α** for the entire dataset. We choose the value of **α** that results in the highest accuracy.

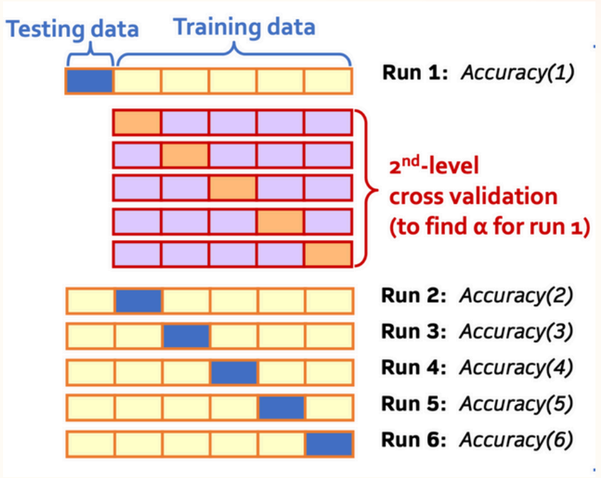


Figure 6: Two Level Cross Validation[12]

[12]Tantum, Stacy. "Brain Computer Interface Movement Decoding Lecture Slides" Duke University. Apr 2024.

Movement Decoding

Movement Decoding - Overview

At this point, we have everything we need to start classifying our EEG data. We have two types of datasets: **imagined** and **overt**, each with 240 observations (120 for each class) of 204-dimensional vectors. Since our dataset is **high dimensional** and the number of features outnumber the number observations, we choose **SVM** as our classifier. We'll be using a linear kernel with our SVM as a baseline. We want to **regularize** this SVM classifier in order to avoid overfitting, but its performance depends on using the optimal regularization strength. Thus, we will use **two-level cross validation** to find the optimal regularization strength. Using this hyperparameter, we can then measure the performance of our classifier.

We will run the following simulations:

- **Same-Train**: Train on overt dataset, Test on overt dataset
- **Same-Train**: Train on imagined dataset, Test on imagined dataset
- **Cross-Train**: Train on overt dataset, Test on imagined dataset
- **Cross-Train**: Train on imagined dataset, Test on overt dataset

For the Same-Train cases, we will report the cross-validated performance. For the Cross-Train cases, we will find apply cross-validation to select the regularization parameter then train on the entire training set before testing on the test set and report the accuracy.

Simulations

Data Visualization

Visualizing our data - Brain Activity Map

Before we present the results of our classification, it is insightful to visualize our data. Here, we calculate the magnitude of each electrode's 2-channel data and create a heatmap on top of the electrode position data. The visualizations below can be viewed as the top of a subject's head (facing up). The first 10 observations of the overt dataset for class *H0* is shown below (*Figure 7*). We also calculated the average observation vector for all 120 observations for each class and visualized it below (*Figure 8 & 9*). It is interesting to note that although we do not know which class corresponds to "left"/"right", we can infer from the **contralateral nature** of the brain that class *H0* is "right" and class *H1* is "left" as we expect **brain activity to be strong on the opposite side** of the user's decision.

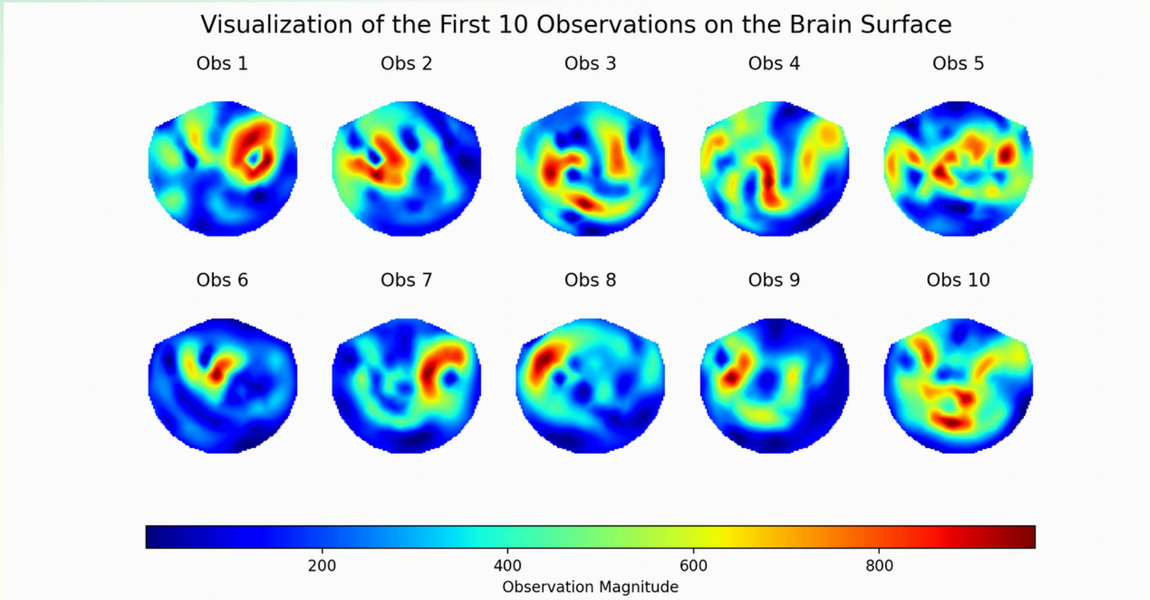


Figure 7: First 10 Observations of Overt H0 Data

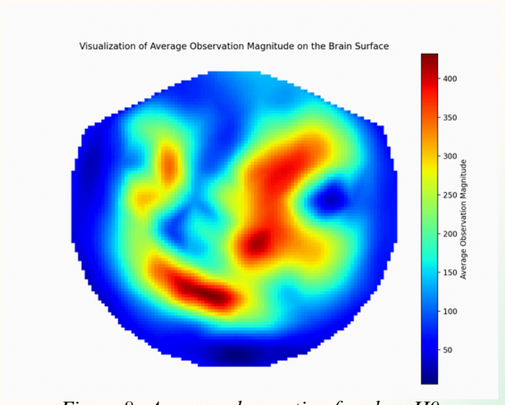


Figure 8: Average observation for class H0

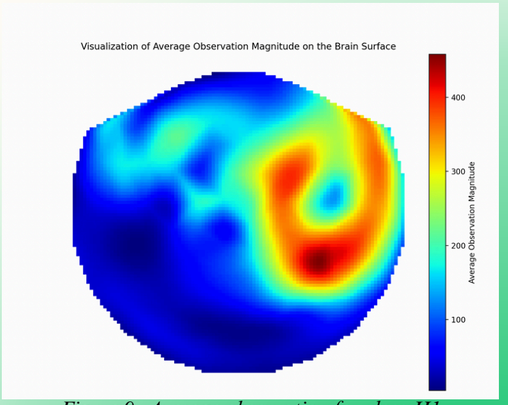


Figure 9: Average observation for class H1

Two Level Cross Validation Results

Cross Validation Single Level - Imagined

The image below (*Figure 10*) visualizes the **magnitude of the channel weights** on the brain's surface of the SVM classifier trained from the first fold of two level cross validation. A larger magnitude (regions in red) indicates that the electrodes in that region carry s**trong directional information** required for classification. We can see that the most important channels for this classification task are in the lower right region of the brain. A stem plot for the signed weights for every channel, along with the 6 dominant channels, are also shown below (*Figure 11*). These weights indicate which channels matter the most for SVM in determining the hyperplane.

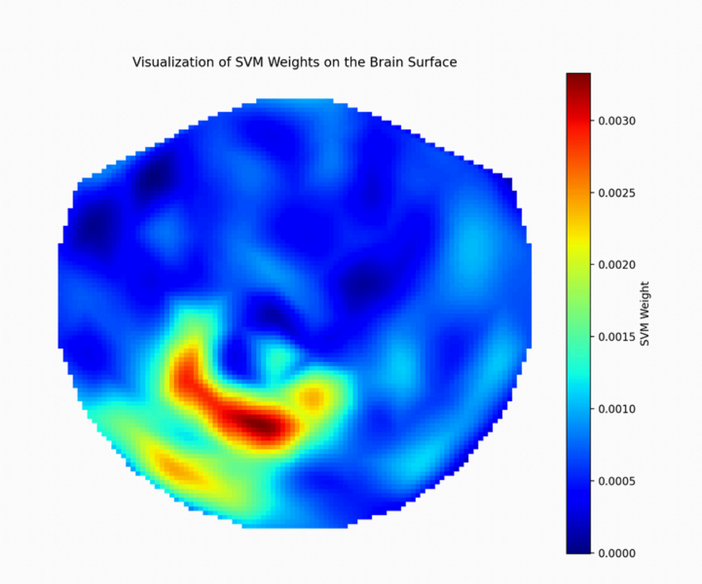


Figure 10: Electrode Weights for SVM (Fold 1)

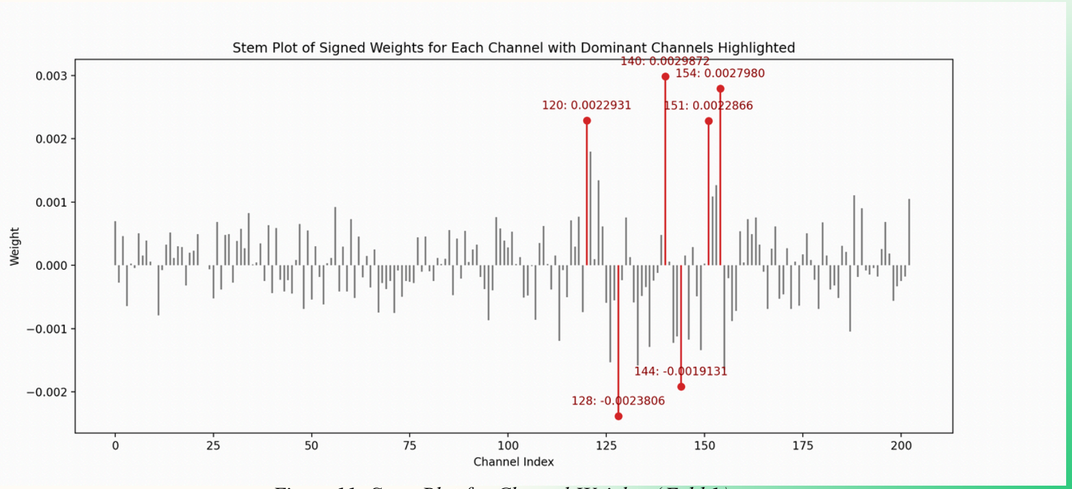


Figure 11: Stem Plot for Channel Weights (Fold 1)

Two Level Cross Validation Results

Cross Validation Single Level - Overt

The magnitude of channel weights and their corresponding stem plot for overt data are shown below (*Figure 10 & 11*). It is interesting to note that there are slightly more regions of the brain that show activity than the imagined EEG dataset. This makes sense since there is **stronger brain activity** when moving your hand as compared to simply thinking about moving your hand. The EEG signals from the overt dataset are inferred to be stronger than the EEG signals from the imagined dataset. As a result, we expect our classifier to **perform better** when trained on overt data as opposed to imagined data.

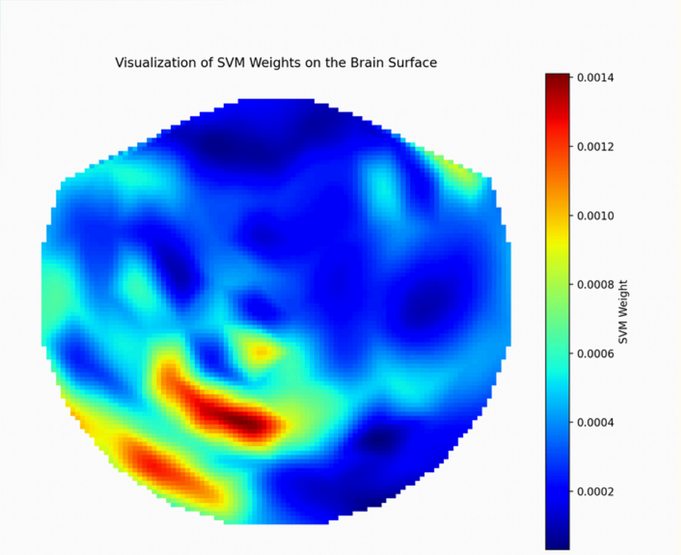


Figure 12: Electrode Weights for SVM (Fold 1)

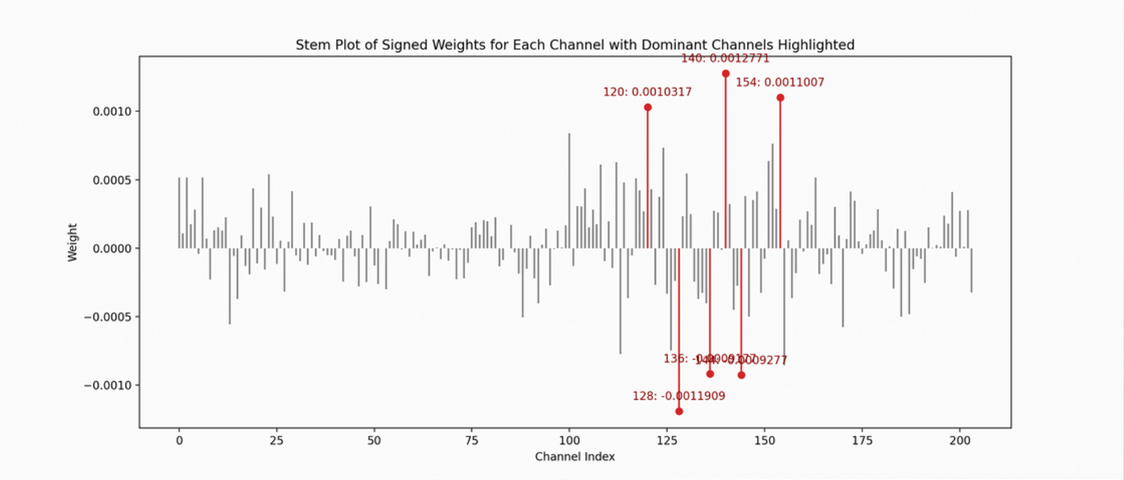


Figure 11: Stem Plot for Channel Weights (Fold 1)

Two Level Cross Validation Results

Cross Validation All Levels - Imagined

After running two level cross validation on the imagined dataset, we calculate the accuracy of the SVM on each testing fold along with the average accuracy (*Figure 12*). We also generate the ROC curve for each fold and determine the average ROC (*Figure 13*). We can see that the individual folds result in accuracies that differ from the average. For example, any individual fold has an accuracy from **0.85** to **0.95**, while the average accuracy was **0.89**. The same is true for the ROC curves. Any individual ROC curve looks noticeably different from the average ROC curve.

We can see that trusting an individual fold can be misleading. We require a more **reliable** picture of how our model might perform under different conditions. This highlights the benefit of using **two-level cross validation**. Since we are working with a limited dataset, we need a reliable and robust way to measure the **performance** and **variation** of our classifier across observations. By splitting our data into subsets, we artificially create "unseen" data. By using a nested cross validation structure, we make effective use of the data available to generate **statistically significant** results.

Fold	Accuracy
1	0.95
2	0.95
3	0.93
4	0.88
5	0.85
6	0.85
Avg	**0.89**

Figure 12: Cross Validated and Average Accuracies - Imagined

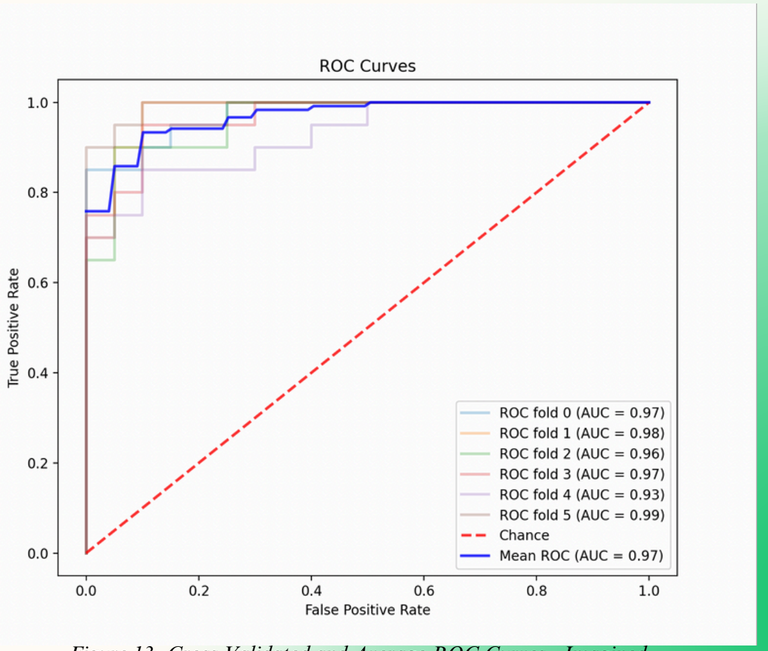


Figure 13: Cross Validated and Average ROC Curves - Imagined

Two Level Cross Validation Results

Cross Validation All Levels - Overt

The accuracies and ROC curves for the overt dataset using two fold cross validation are shown to the right (*Figure 14 & 15*). We see that the overt data results in a higher average accuracy of **0.95** and an average ROC curve with a higher AUC than imagined data. This confirms our hypothesis that our SVM classifier **performs better on overt data** than imagined data.

Overt movement data typically produces clearer and more robust EEG signals because **actual movement generates stronger brain activity**, which are easier to detect and differentiate. Imagined movements, on the other hand, generate subtler EEG signals, often mixed with more noise and other thought processes, making the signal less distinct. This difference in signal clarity results in overt data having **better separability** of classes, causing a higher accuracy.

Fold	Accuracy
1	0.95
2	0.98
3	0.98
4	1.00
5	0.85
6	0.95
Avg	**0.95**

Figure 14: Cross Validated and Average Accuracies - Overt

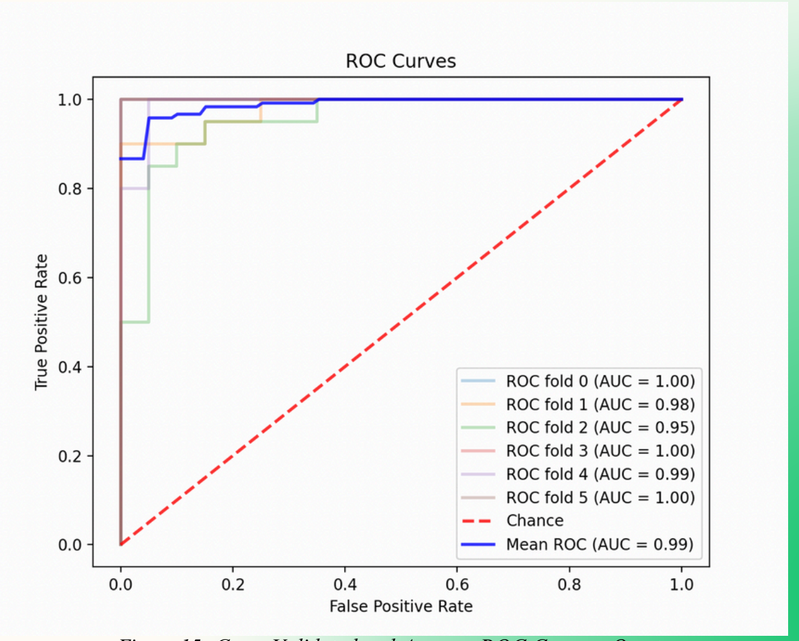
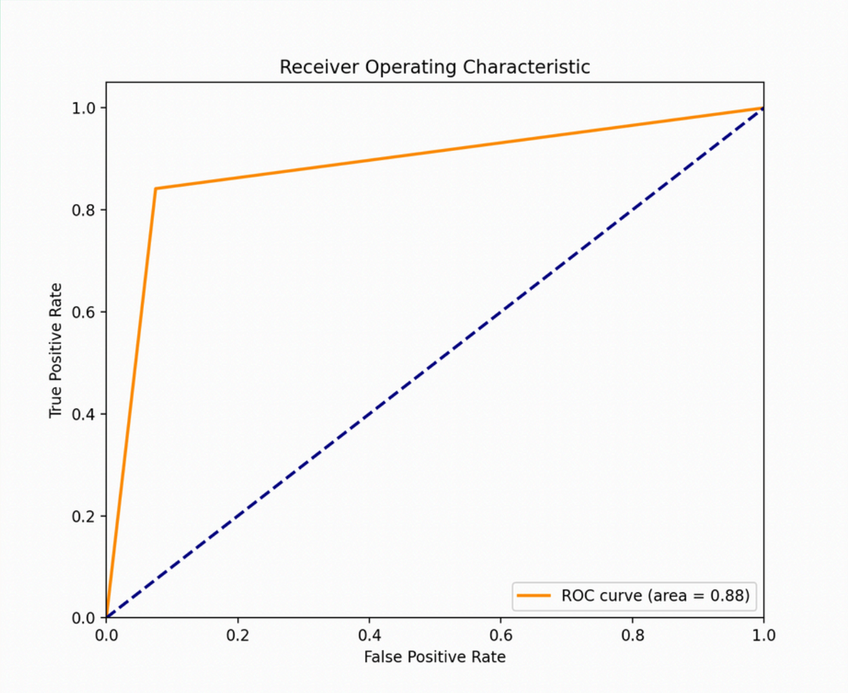


Figure 15: Cross Validated and Average ROC Curves - Overt

Classification Performance

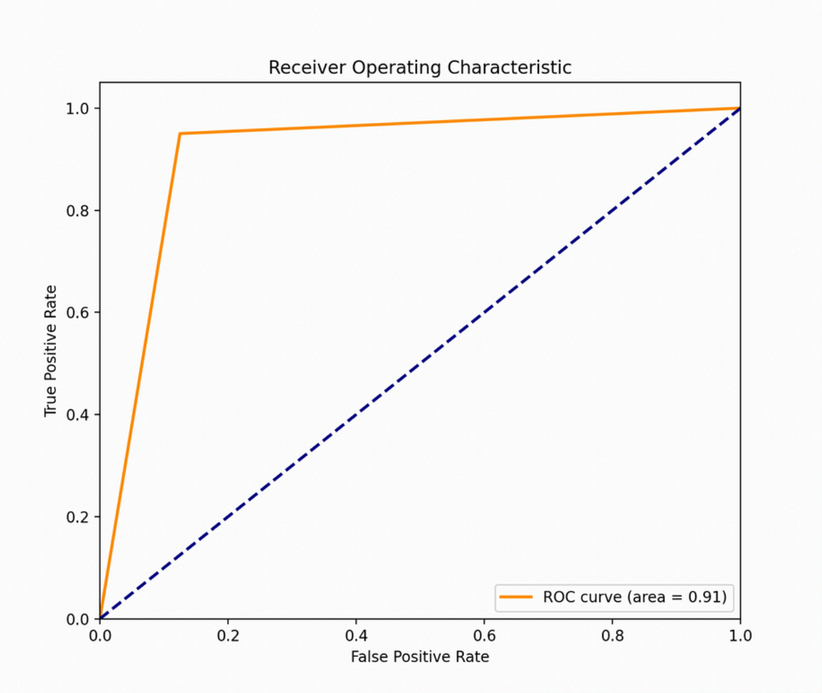
Cross-Train Results - ROC + Accuracy



Train on Overt, Test on Img.

Accuracy: 0.88

Figure 16: ROC + Accuracy (Train on Overt, Test on Img.)



Train on Img. Test on Overt

Accuracy: 0.91

Figure 17: ROC + Accuracy (Train on Img., Test on Overt)

Classification Performance

Cross-Train Results - Observations

The area under the ROC curve (AUC) for training on overt and testing on imagined data is **0.88**, while the AUC for training on imagined and testing on overt data is **0.91**. These are both good results, indicating a strong classifier in both scenarios, with the latter being slightly better.

Training on overt and testing on imagined (AUC=0.88): We are training with data from a more concrete task (overt movement) and testing it on a more abstract task (imagined movement). The fact that the accuracy is high (0.88) suggests that the patterns learned from the overt movements generalize fairly well to imagined movements. However, there may be a drop compared to the second scenario because the overt movement likely generates stronger and more distinct EEG signals, which might not capture the subtler nuances of imagined movement.

Training on imagined and testing on overt (AUC=0.91): When the model is trained on imagined data and tested on overt data, the accuracy increases to 0.91. Imagined tasks might produce more variable EEG patterns due to the nature of thought processes that are less consistent than physical movements. Training on this noise-inclusive data could lead the SVM to identify more robust decision boundaries, hence it performs slightly better when tested on the clearer signals of overt data.

Classification Performance

Cross-Train Results - Different Kernels

Thus far, we've been using a **linear kernel** with our SVM. This means that our SVM finds a hyperplane in the ambient dimension of our feature. This works well if our data is linearly separable in that dimension. However, the class clusters in our dataset can be non-linearly separable as well. In these scenarios, SVM makes use of **the Kernel Trick**[13] to map the observations from the ambient dimension to a higher dimension. The main idea is that projecting the data to a higher dimension can make it linearly separable. SVM can then find a hyperplane in the higher dimension to separate the data.

We tested two kernels with our SVM, polynomial and Radial Basis Function (RBF). Our initial hypothesis is that using these kernels should improve performance since they add complexity to our model (ability to separate non-linear data). The cross-train results are shown below (*Figure 18*).

Kernel	Train on Overt, Test on Img.	Train on Img. Test on Overt
Linear	Accuracy: 0.88	Accuracy: 0.91
Polynomial	Accuracy: 0.79	Accuracy: 0.79
RBF	Accuracy: 0.79	Accuracy: 0.89

Figure 18: Cross-Train Accuracies with different Kernel Functions

We can see from the simulation results that both the Polynomial and RBF kernels actually result in **decreased performance**. Projecting our data from an already high dimensional space (our EEG data is 204-dimensional) to an even higher dimension might not be advantageous and can lead to **overfitting**. Despite being the simplest model, the linear kernel offers better generalization across data and does not overfit.

[13]Hsu, Chih-wei & Chang, Chih-chung & Lin, Chih-Jen. (2003). A Practical Guide to Support Vector Classification Chih-Wei Hsu, Chih-Chung Chang, and Chih-Jen Lin.

Conclusions

Conclusions

Interpreting the Results

Visualization and Contextual Knowledge - Domain experts can use contextual knowledge to interpret patterns found in visualizations. For example, when visualizing the SVM Channel weights on the brain's surface, we saw regions of the brain that were more important for the classification task at hand. Certain brain regions are known to be more involved in motor tasks and intentions. For instance, the primary motor cortex is critical for initiating voluntary movements, so you would expect higher SVM weights in regions overlying the motor cortex when the task involves distinguishing left vs. right hand movements. A domain expert could also see that, when visualizing the input data on the brain's surface, there is a lateralization effect that is expected - each hemisphere controls movements on the opposite sides of the body. This can help us infer the mapping of label to class (*H0*/*H1* to "left"/"right")

Performance Trends in Cross-Data Training/Testing: Training the classifier on more variable (noisy) data and testing on less variable (clear) data seems to yield better generalization. This may be because the variability in the training data forces the SVM to find decision boundaries that are more robust to noise and variation, which are effective when applied to the clearer testing data. If I could design the training data to have a higher or lower signal-to-noise ratio (SNR) than the testing data, I would likely design the training data to have a slightly lower SNR. This approach, often referred to as training with noise, is known to potentially increase the robustness of the classifier. A classifier that can handle noisy training data should be able to generalize better to new data, which might not be as noisy. This approach would help in avoiding overfitting to the noise in the training set and improve the model's ability to generalize to unseen data.

Complexity is not always better: The use of RBF and polynomial kernels can be beneficial in the presence of non-linear class boundaries. But their effectiveness depends on the dimensionality and specific structure of the dataset at hand. For EEG data, which is already in a high-dimensional space, the additional complexity introduced by these kernels must be carefully balanced against the risk of overfitting, as our results indicate that the simplest model does not always perform worst, and sometimes a linear approach is more appropriate

Conclusions

Lessons Learned

Factors that may impact classification accuracy:

- **Signal Quality**: The signal-to-noise ratio of the EEG data can significantly impact classification.
- **Model Complexity**: The balance between a model that is complex enough to capture nuances in the data without overfitting.
- **Training/Testing Split:** Splitting our data into stratified folds for use in Cross Validation is important. Using folds with biases in the data can generate misleading results.

Limits with our approach/Possible Improvements:

- **Static Model:** The current SVM model does not account for temporal dynamics in EEG data, which could be informative.
- **Enhanced Feature Extraction:** Utilizing more advanced signal processing techniques to better capture the informative features of the EEG data (e.g. using magnitude/phase instead of x/y gradients).
- **Cross Validation for Hyperparameters:** I did not perform Cross Validation for the gamma parameter when using the Polynomial or RBF kernels (automatically set by *sklearn)*. Thus, the decrease in performance may be misleading, but I believe even with Cross Validation, the results would have been the same.

Anything unique you have done to improve/validate your classifier's accuracy/efficiency

- **Different Kernels:** I tested the SVM classifier with RBF and polynomial kernels in hopes that increased complexity would improve performance.
- **PCA Dimensionality Reduction:** I initially attempted to reduce the dimensionality of the dataset using PCA and measured accuracy using the linear, polynomial and RBF kernels. PCA seemed to always have a negative effect on cross-validated performance, so I did not include it in my results (different kernels also hurt performance but it was easily interpretable).

References

References

“If I have seen further, it is on standing on the shoulder of giants.” -Issac Newton

Background Information

[1]Shih, Jerry J et al. “Brain-computer interfaces in medicine.” Mayo Clinic proceedings vol. 87,3 (2012): 268-79. doi:10.1016/j.mayocp.2011.12.008

[2]Altman, N., Krzywinski, M. The curse(s) of dimensionality. Nat Methods 15, 399–400 (2018). https://doi.org/10.1038/s41592-018-0019-x

[3]Hsu, Chih-wei & Chang, Chih-chung & Lin, Chih-Jen. (2003). A Practical Guide to Support Vector Classification Chih-Wei Hsu, Chih-Chung Chang, and Chih-Jen Lin.

[4]Scikit-learn: Machine Learning in Python, Pedregosa et al., JMLR 12, pp. 2825-2830, 2011.

[5]Harris, C.R., Millman, K.J., van der Walt, S.J. et al. Array programming with NumPy. Nature 585, 357–362 (2020). DOI: 10.1038/s41586-020-2649-2.

[6]J. D. Hunter, "Matplotlib: A 2D Graphics Environment", Computing in Science & Engineering, vol. 9, no. 3, pp. 90-95, 2007.=

[7]A. Basu, C. Walters and M. Shepherd, "Support vector machines for text categorization," 36th Annual Hawaii International Conference on System Sciences, 2003. Proceedings of the, Big Island, HI, USA, 2003, pp. 7 pp.-, doi: 10.1109/HICSS.2003.1174243.

[8]Salunkhe, Vivek. “Support Vector Machine (SVM).” Medium, Medium, 23 July 2021, medium.com/@viveksalunkhe80/support-vector-machine-svm-88f360ff5f38.

[9]Tantum, Stacy. “Brain Computer Interface Movement Decoding Lecture Slides” Duke University. Apr 2024.

[10]Hoerl, Arthur E., and Robert W. Kennard. “Ridge Regression: Biased Estimation for Nonorthogonal Problems.” Technometrics, vol. 12, no. 1, 1970, pp. 55–67. JSTOR, https://doi.org/10.2307/1267351. Accessed 28 Apr. 2024.

[11]Tantum, Stacy. “Brain Computer Interface Movement Decoding Lecture Slides” Duke University. Apr 2024.

[12]Tantum, Stacy. “Brain Computer Interface Movement Decoding Lecture Slides” Duke University. Apr 2024.

[13]Hsu, Chih-wei & Chang, Chih-chung & Lin, Chih-Jen. (2003). A Practical Guide to Support Vector Classification Chih-Wei Hsu, Chih-Chung Chang, and Chih-Jen Lin.

SVM

[8]Salunkhe, Vivek. “Support Vector Machine (SVM).” Medium, Medium, 23 July 2021, medium.com/@viveksalunkhe80/support-vector-machine-svm-88f360ff5f38.

Ridge

[10]Hoerl, Arthur E., and Robert W. Kennard. “Ridge Regression: Biased Estimation for Nonorthogonal Problems.” Technometrics, vol. 12, no. 1, 1970, pp. 55–67. JSTOR, https://doi.org/10.2307/1267351. Accessed 28 Apr. 2024.

Visualizations from Other Sources

[8]Salunkhe, Vivek. “Support Vector Machine (SVM).” Medium, Medium, 23 July 2021, medium.com/@viveksalunkhe80/support-vector-machine-svm-88f360ff5f38.

[11]Tantum, Stacy. “Brain Computer Interface Movement Decoding Lecture Slides” Duke University. Apr 2024.

Toolboxes/Packages

[4]Scikit-learn: Machine Learning in Python, Pedregosa et al., JMLR 12, pp. 2825-2830, 2011.

[5]Harris, C.R., Millman, K.J., van der Walt, S.J. et al. Array programming with NumPy. Nature 585, 357–362 (2020). DOI: 10.1038/s41586-020-2649-2.

[6]J. D. Hunter, "Matplotlib: A 2D Graphics Environment", Computing in Science & Engineering, vol. 9, no. 3, pp. 90-95, 2007.

Collaborations

Collaborations

“Great things in business are never done by one person” -Steve Jobs

I worked entirely independently for this project. I don’t know anyone else in the class and I found myself very invested in this project. In absence of other collaborators, I relied on the following resources:

- StackOverflow: debug code, look for specific functionality (e.g. plot a heatmap)
- Library Documentation: APIs for the libraries used in this project to understand functionality of specific methods
- ChatGPT: debug code (e.g. find the bug in model)