Data Analysis and Model Classification

Mini-project 2

Authors:
Julie Laurent
Alice Leydier
Yara-Maria Proust

Group ID : 2

November 18, 2017

ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

1 Introduction

Without even noticing it, one person is able to make series of motions going from a simple movement of a finger to walking or running. All these daily events, are in fact the result of complex transfers of electrical activity all along the body. Indeed, it is the brain that sends a certain signal for the body to move. In particular, Event Related Potentials (ERPs) are electrical brain waves generated as reaction to sensory, cognitive or motor events. ERPs can be detected by means of noninvasive way as Electroencephalogram (EEG), which allows the recording of electrical activity in the brain with electrodes placed along the scalp. In this project, ERPs were recorded on participants watching a cursor moving towards a specified target. Sometimes, the cursor moved on the opposite way, creating an error-related ERP in the patient. Based on this experiment, a file was generated, containing the 2400 features and 648 samples (observed cursor movements). This file contained also the labels, which was containing only "1" (correct movement) or "0" (erroneous movement). Thus, the data could be separated into two classes: Class A representing the correct movements and Class B regrouping the erroneous ones. The first step was a supervised learning approach (as the collected data were here labeled) to explore the data. After identifying the features having the highest discrimination between the two classes, a feature threshold could be computed and its performance was tested. A generalization was then made in order to see if this classifier was powerful on a new dataset. Linear and Quadratic Discriminant Analysis classifiers (LDA and QDA classifiers) were performed, and a training and testing errors step was undertaken. Moreover, a k-fold cross validation followed by the creation of a confusion matrix was needed in order to estimate the performance of the models. The best number of features to train a model (hyperparameter) then had to be chosen. This was done by cross-validation and nested cross-validation, to estimate the true performance, as simple cross-validation is a biased estimation of the true error. The same steps were then performed to select the classifier type. Several possible information extraction methods exist, notably by combining features either by changing feature selection method (Forward features selection (FFS) or *Fisher* ranking) or by engineering new features (Principal Component Analysis (PCA)). They were tested, in order to compute their combination with the classifier type. Finally, the statistical significance of the mean test errors across outer folds of the previous nested cross-validations were computed to select the best model and its associated hyperparameters.

2 Methods

Data exploration In order to understand the data given and compute a model to separate the error-related ERPs from the correct ones, the distribution of the features was observed, using histograms where the distributions of Class A and B were superposed for each feature. To observe the median of these distributions, their boxplots were computed, using Notch to visualize the 95% confidence interval. If the confidence intervals were overlapping, no significant difference between classes could be proved, otherwise, the difference was significant.

Features selection To objectively validate a real difference of distributions for Class A and B in the features, a two samples t-test was computed, with the null hypothesis stipulating that distributions from both classes had equal means, with a 95% confidence interval (meaning the alpha-level was 5%). As this test assumes features with independent samples following normal distributions, only features having normal distributions were selected (Lilliefors test). In addition, different variances for the classes were specified in the t-test. If the obtained p-value was smaller than alpha, the null hypothesis was rejected: the classes had statistically significant different distributions. As the lower the p-value, the more significant the difference in the mean values is, the features with the lowest p-values were chosen

to build the model of discrimination between the two classes. For this last part, the fact that the alpha-level needed a correction was taken into account (Bonferroni correction), to counteract the problem of multiple comparisons. Multiple testing indeed requires a correction because the more population one tests, the more chance there is to get a significant difference in a test.

Features thresholding The next step was to use the best feature found before to compute a threshold allowing the separation of samples between both classes, as they would theoretically be on different sides of the threshold. To do so, the class error and the classification error were calculated. As the class error takes into account that the classes do not have the same number of samples (ratio: Class A/Class B = 516/132 in this project), it was used to calculate the more accurate threshold, by searching for the minimal class error:

$$\frac{1}{2}\frac{\text{number of misclassified samples of Class A}}{\text{number of total samples of Class A}} + \frac{1}{2}\frac{\text{number of misclassified samples of Class B}}{\text{number of total samples of Class B}}$$

Generalization To assess the performance of the threshold classifier elaborated before, the dataset was separated randomly into a training and testing set. The percentage of training samples was changed, from 10% to 90% by steps of 5%. The training set was used to determine the threshold as previously done, while the testing set assessed this threshold efficiency, by looking at the class error metrics on the testing set.

LDA/QDA classifiers Here, only a subset of 60 original features was used, selected by taking one out of five features, until the 300^{th}. *Diaglinear, linear* and *quadratic* discriminant classifiers were computed. The same subset used for the training of the classifiers was "tested", by predicting the labels corresponding to a feature. Then, the class error was calculated for each classifier. Prior probabilities were introduced, as they represent the chance of having correct or erroneous movements before training and testing. By default, the MATLAB function, used to build the classification decision tree, sets prior probabilities as 'empirical', corresponding to the relative frequencies of the labels. Thus, new classifiers were computed with 'uniform' prior probabilities, meaning all class prior probabilities were equal to $\frac{1}{K}$, with K the number of classes, thus equal to $\frac{1}{2}$. Once again, the class error was computed and compared between classifiers with both prior probabilities.

Training and testing errors A difference between training and testing sets was taken into account, as the testing set had to be an *unseen* dataset. From the subset previously used and after randomization, a separation between two different sets was made, *Set1* (training set) and *Set2* (testing set), in order to have the same number of samples in each set (324 samples each) without any influence in the choice of samples. Then, an estimation of the training and testing errors was done. These are the class errors of a set with a classifier trained respectively on the same set, or on another set. To do so, testing was done two times on *Set1*. The first time, the three classifiers (*linear, diagquadratic and quadratic*) were trained on the same *Set1* and gave the training error, while the second time, the classifiers were trained on the *Set2*, giving the testing error. Then, the training and testing errors of *Set2* were computed.

Cross-Validation (CV) By splitting the whole dataset in two (training and testing sets), some information was lost, which was problematic due to the small number of samples. To counteract this issue, a 10-fold CV approach was used. The data was separated into 10 subsets of equal size ($\pm$ 1). Nine subsets were used for the training and the remaining set was used for the testing. This CV partition was based on the labels in order to obtain similar ratios of ClassA/ClassB within folds. For each *linear, diaglinear and diagquadratic* classifier, with different training and testing sets, the class error (testing error) was computed.

To measure how stable was the performance of all classifiers, the CV error (mean of test errors across folds), as well as its standard deviation (SD), were computed. Each classifying method had a different number of parameters to be estimated and thus their complexity differed. To assess the robustness of the models, the number of parameters for each classifier was compared to the number of training samples. Then, a *"leave-one out"* CV was generated, where the number of folds is equal to the number of samples, leading to one-sample folds. Both CV types were then repeated with four different partitions of the folds. Finally, the variability of the performance between classifiers trained with *Set1* and *Set2* was studied.

Confusion Matrix A confusion matrix was computed to measure the quality of the selected classifier. Its particular utility is to make it easy to see if the system is confusing between the two classes, as in some cases, it can be worse to have False Negative or False Positive errors.

Cross-validation for hyperparameters optimization After this introduction to model selection, an optimization of the hyperparameters was done by choosing the best number of features to use. To optimize this hyperparameter, a 10-fold CV was done using a *diaglinear* classifier. A *Fisher*-based features selection was done inside the CV loop (computed only on the training set), and testing and training errors were calculated, by increasing the number of used features each time, starting with the best one selected by *Fisher*, then the two best ones, and so on. *Fisher* measures how well a feature has within-class similarities and different values for different classes, and order them accordingly. A plot of the training and testing errors of each number of features and the mean error across folds was then computed. The same computation was done, using a *Pearson correlation* for features selection. *Pearson* method takes into account the strength of linear dependence between two variables, here the values of a feature and the sample labels. Finally, 648 normal random label vectors were computed and 1000 10-fold CVs were computed in the same way as before, using features selected by *Fisher* ranking. The mean test error of the random classifier across folds was computed for each CV repetition, as well as the minimum error over these 1000 test errors.

Nested Cross-validation for performance estimation The error obtained with simple CV is biased as it only allows to choose the best number of features amongst others without assessing the model's performance based on this particular set of features chosen. This is why a nested CV had to be made, where the performance of the selected model was tested with an outer CV, while hyperparameter selection was performed in an inner CV inside each fold of the outer CV. Thus, an accurate estimation of the test error on a new unseen dataset could be obtained. Nested CV was implemented with *Fisher*-based features selection, using a *diaglinear* classifier again. A 10-folds were chosen both for the inner and the outer CVs. The choice of 10 inner folds enabled consistency with the simple 10-fold CV done before for the hyperparameter optimization. 10 outer folds were chosen to have a sufficient number of values to assess the variation of the model. However, no more than 10 were considered due to time-consuming programs. Features selection was computed once inside the outer CV and another time inside the inner CV because it had to be based on the training set, changing in each outer and inner fold. The mean validation error across inner folds ("test" error for each hyperparameter value) was calculated and the lowest value (optimal validation error) enabled the selection of the best model. This model was then used to build the classifier on the outer-fold training set and assess the outer training and testing errors. A comparison of the median errors of the hyperparameters (median value of the optimal validation errors obtained after each inner CV) to the hyperparameter found in the simple CV (previous section) was done. The median was considered here instead of the mean to exclude the impact of the outliers. Finally, a boxplot of the distributions of optimal training (low-

est train error after each inner CV), optimal validation error and test error (obtained in the outer CV) was plotted. The same process was done again by changing the classifier type (between *diaglinear, linear* and *diagquadratic*) as a hyperparameter on top of the number of features.

Principal Component Analysis (PCA) Use of PCA (unsupervised technique) allows the projection of the original data into a new features space defined by Principal Components (PCs), which generate uncorrelated features corresponding to linear combinations of the original ones, defined to maximize the variance of the original data. These uncorrelated features were then used as training set for the clustering. Covariance matrices of the original data and the data projected on PCs were computed and plotted. A comparison between the diagonal elements (variances) of these matrices and between the off-diagonal elements (covariances) were done. The variance corresponds to the variability with respect to classes inside a feature, while the covariance is a measure of the correlation between features. To choose the number of PCs to be retained (being a hyperparameter), the result of the cumulative variance with respect to the total variance was plotted. Before applying PCA, a normalization was made on the training subset, for the same reasons than before: the testing or unseen set could not be modified. PCA was computed on the normalized training set and incorporated into the CV instead of features ranking. The first 60 PCs were selected and treated as hyperparameters, while the classifier type was fixed to *diaglinear*. The testing set was transfered into the new PCs space as following:

$$testSet_{PCA} = \frac{testSet_{original} - \mu}{\sigma} * coeff$$

where μ and σ are respectively the mean and the SD of the training set while $coeff$ represents the PC coefficients obtained with the PCA training set.

Forward features selection (FFS) By using the *Fisher*-based algorithm, only the individual discriminability of features was considered, meaning that two correlated features could have been selected, leading to a surplus of complexity without any additional information. FFS remedies to that by selecting a subset of relevant features to use in model construction. It allows a simplification of the model leading to shorter training times, avoids the curse of dimensionality, and enhances generalization by reducing over-fitting. *Fisher* score is a filter univariate method as it does not need to build a model to evaluate the features, while FFS is a multivariate wrapper method as features are selected and evaluated at the same time with the help of a model. FFS starts by selecting the best performing (in terms of class error) single feature, then adds a second feature that leads to the best joint performance, and a third, and so on. The MATLAB function used performs itself the 10-fold inner-loop CV to determine the optimal number of features, and selects a subset of features that best predict the labels, by sequentially selecting features until there is no improvement in the chosen criterion. A *diaglinear* classifier was used, and the model selection was done as before in a 10-fold outer CV. Finally, the mean test error across outer folds was computed with the number of features given by the minimal validation errors.

Combining the features engineering and selection with classifier types As PCA is an unsupervised technique, combining it with *Fisher* features ranking enables to add the criterion of class discrimination within the selected features. It was implemented within the inner loop of a nested CV (10 outer and inner folds), using a *diaglinear* classifier. The PCA was done before the ranking, and after normalization on the training set. The best number of PCs and features according to inner CV was then used as hyperparameters in the outer CV. A FFS was combined to a classifier selection inside another nested CV. Inside each outer fold, the FFS was repeated for each classifier type to choose the best features and the best classifier.

Statistical Significance To compare models' accuracy to the random level of 50% (2-class problem), the mean test error and its SD across outer folds were calculated. A t-test with the null hypothesis that the test error values came from the normal random distribution with a mean of 50% was done. To determine if the test error values were normally distributed (as required by a t-test), a histogram was plotted to visualize the normality, and a Kolmogorov-Smirnov test was performed, after normalization of the test error values. If the null hypothesis was rejected, a non-parametric statistical test (Wilcoxon signed rank test) was performed.

3 Results

Statistical significance By looking at the feature distributions (Fig.1), good (very distinct distributions for both classes) and bad (similar distributions for both classes) features could be identified. Feature 305 was selected as a bad feature (p-value = $109.05 > 0.05$) and feature 681 as a good one (p-value = $6.19e^{-23} < 0.05$). Features had a normal distribution, as required for a t-test.

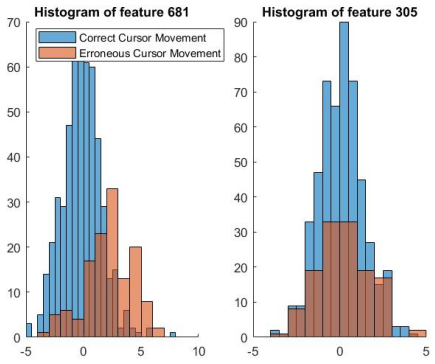

Figure 1: Histograms of a good feature (feature 681, with the minimal p-value) and a bad one (feature 305). Class A distributions are shown in blue and Class B distributions in red.

Looking at the boxplots (Fig.2), medians from both classes were not equal for the good feature (681) whereas almost no difference was observed for a bad feature (305). Moreover, the Notches on the good feature did not overlap, confirming a significant difference between classes for this feature.

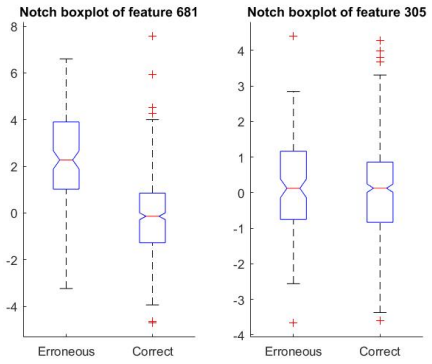

Figure 2: Boxplots of a good feature (681), and a bad one (305). Class A distributions are on the right and Class B distributions on the left.

Features thresholding After selection of the "best" feature (681, minimal p-value), a *linear* classifier was built based on it. To do so, the minimal errors were computed (Fig.3): respectively 0.2185 and 0.1497 for the class and classification errors. Class error was kept for coherence with the dataset (unequal number of samples per class), and the optimal threshold found was 0.93.

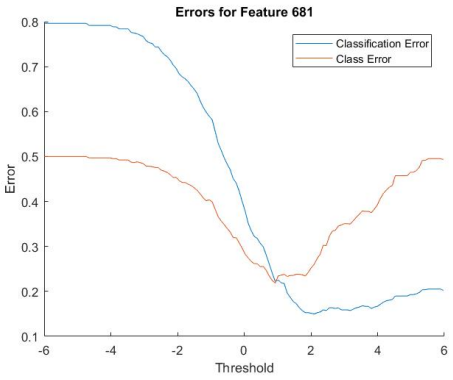

Figure 3: Classification and class errors across different threshold values of feature 681. Classification error is shown in blue and class error in red.

Feature 681 was plotted vs. another good feature 531 (p-value = $1.51e^{-21} < 0.05$) (Fig.4). The optimal threshold was added to this graph in order to compare the results with the original class labels. Most of the samples were on the correct side of the threshold, but still a lot of mis-classifications were observed (error: 0.4522).

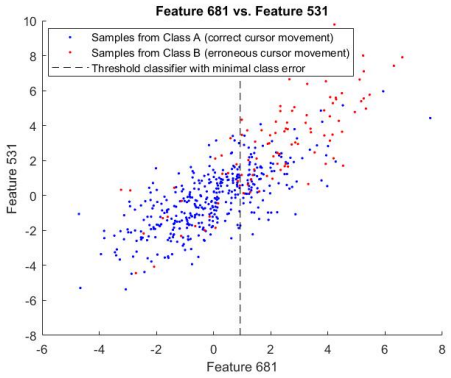

Figure 4: Scatter plot of feature 681 vs. 531. Class A samples are shown in blue and Class B samples in red. The threshold is represented by the dashed vertical line.

Generalization Similar ratios were obtained for different splits between train and test sets, as assumed earlier (original ratio: 3.79 ; training set ratio: 3.84 ± 0.28 ; testing set ratio: 4.03 ± 0.45). The class error was computed on both sets and plotted for each percentage split (Fig.5). As the training set percentage got larger, the training error was quite stable. At the same time, the testing error decreased a lot and rose very fast towards the end. Minimal testing error was found when the dataset was split with 80% to 20%.

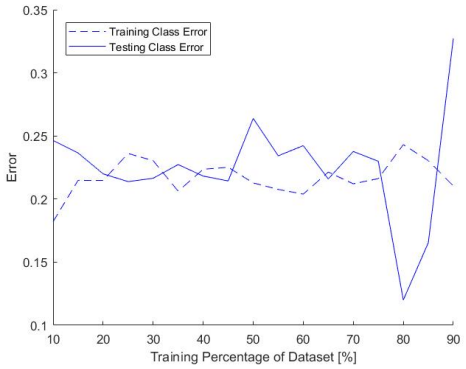

Figure 5: Training and testing class errors, for each percentage split. Training is shown with a dashed line and testing with a plain line.

LDA/QDA classifiers By looking at the class errors across classifiers (Fig.6), the lowest class error for the *quadratic* (0.0038). On the same figure, comparing the class error with 'empirical' or 'uniform' probabilities, 'uniform' probabilities systematically had a smaller class error for all classifiers except for the *diagquadratic* one. This classifier with prior probabilities set at 'empirical' was the second lowest error with the 'empirical' parameter after the *quadratic* classifier (0.2111).

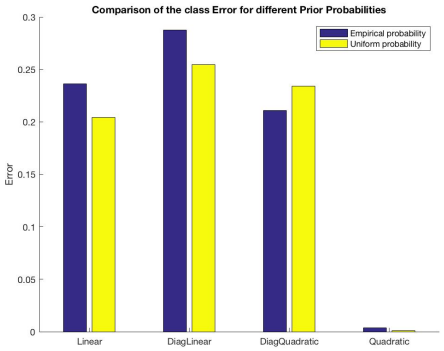

Figure 6: Comparison of class errors for each classifier with 'empirical' (in blue) or 'uniform' (in yellow) prior probabilities.

Training and testing errors Looking at the training and testing errors (without *quadratic* classifier), the training error was smaller than the testing one for each classifier type (Fig.7 and 8). On Fig.7, the classifier with the smallest error was the *diagquadratic* one (*diaglinear*: 0.3379, *linear*: 0.3081, *diagquadratic*: 0.2677). Improvement on training error did not improve testing error.

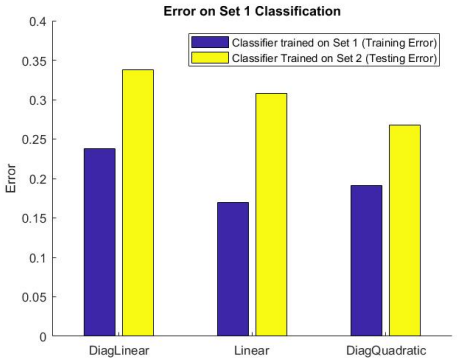

Figure 7: Training (in blue) and testing (in yellow) errors on *Set 1* for each classifier.

Looking at errors on *Set 2* (Fig.8), the smallest testing error corresponded also to the *diagquadratic* classifier (*diaglinear*: 0.3013, *linear*: 0.3137, *diagquadratic*: 0.2663). Finally, by comparing the errors obtained on *Set 1* and *Set 2*, a great variability of performance was observed.

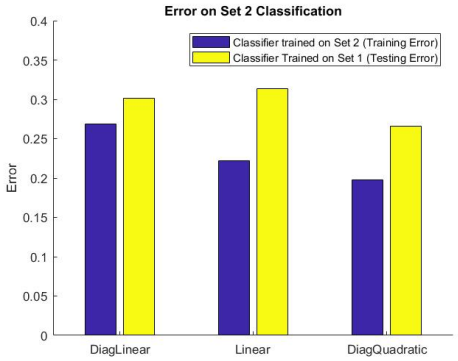

Figure 8: Training (in blue) and testing errors (in yellow) on *Set 2* for each classifier.

Cross-Validation (CV) As shown in Fig.9, the *diagquadratic* classifier obtained the smallest CV error (*diaglinear*: 0.3081 ± 0.0825, *linear*: 0.2835 ± 0.0539, *diagquadratic*: 0.2298 ± 0.0516). Smaller errors were obtained with the *leave-one out* CV (*diaglinear*: 0.2191 ± 0.4140, *linear*: 0.1420 ± 0.3493, *diagquadratic*: 0.1944 ± 0.3961), but SDs were much higher, as shown in Fig.10.

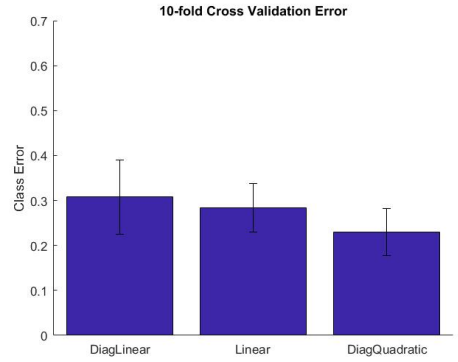

Figure 9: 10-fold cross-validation errors and their SDs for each classifier type.

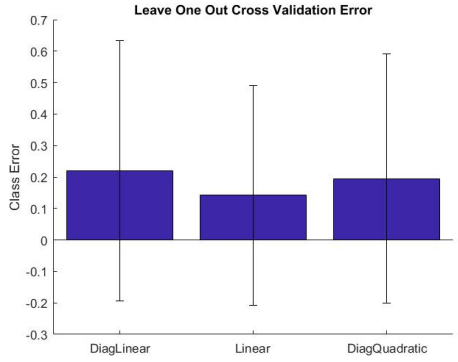

Figure 10: *Leave-one out* CV errors and their SDs for each classifier type.

Then, CV errors and their SDs for the best *diagquadratic* classifier were looked at, with four different partitions. *Leave-one out* CV showed the same error across different partitions with large SDs (mean of errors: 0.1944 ± 0.3931). On the contrary, the 10-fold CV errors showed slight differences in the means, but small SDs. The means were between 0.2379 and 0.2483 and the SDs were between 0.0566 and 0.0860.

The number of parameters for each classifier was calculated: *linear* and *diaglinear* classifiers had 61 parameters, *diagquadratic* 121, and *quadratic* 3661. For the following steps, the *diaglinear* classifier was chosen for its low complexity, also limiting the risk of over-fitting.

Confusion Matrix The computed confusion matrix found was the following :

$$CM = \begin{bmatrix} 45 & 25 \\ 35 & 219 \end{bmatrix}$$

The element in the first row and column represents the *True Positive*, in this case 45 erroneous movements correctly classified as erroneous. The second element of the first row is called *False Negative*, in this case 25 erroneous movements misclassified as correct ones. The element in second row, first column corresponds to *False Positive*, here 35 correct movements misclassified as erroneous ones. The last element in the second row and column is the *True Negative*, here 219 correctly classified correct movements.

Cross-validation for hyperparameters optimization First, a *Fisher*-based features selection was applied. It was seen in Fig.11 that errors diminished a lot until 20 features were used to build the model, where the smallest testing error across folds was obtained (0.1930). With a greater number of features, the training error stabilized and after 50 features it was overlapping with the testing error. As no more change was observed for the two errors beyond 50 features, the maximum number of features used was 60. As expected, the SD was much greater for the testing error. Features were then selected using the *Pearson* correlation. In this case, the train and test errors

had a tendency to increase when more than 13 features were selected to build the model. The order of magnitude of the errors was however the same in both types of features selection.

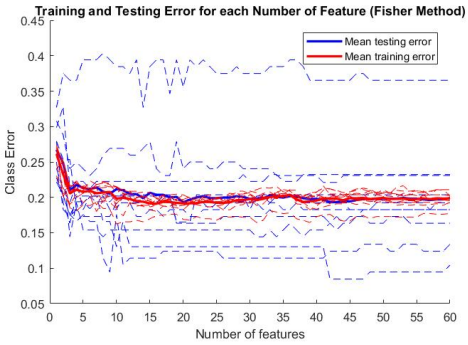

Figure 11: Training and testing errors for each chosen number of features, using the *Fisher* method. The dashed lines correspond to the error values for each fold and the full lines correspond to the mean errors. The color blue refers to the testing errors and red to training errors.

In the random label process, the mean test error across folds was 0.5005. After repeating the process 1000 times, the minimum mean test error found was 0.4880.

Nested cross-validation for performance estimation Across the 10 outer folds, the number of selected features (*Fisher* method) varied a lot: 11, 13, 16, 19, 22, 24, 24, 28, 31, 46 (median: 23). The median error (optimal validation) of those hyperparameters was 0.1967. The mean test error across outer folds was 0.2013 ± 0.0944. Distributions of the optimal training, validation and test errors are shown in Fig.12. The training error was the smallest (median: 0.18949), then the validation error (median: 0.19665) and finally the test error was the largest (median: 0.21641). Comparing the SDs, the training error had the less variations and the testing the most from far.

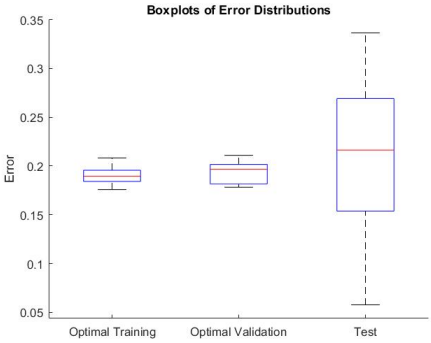

Figure 12: Boxplots of optimal training, validation and test errors.

The classifier choice was added on top of features selection. Across the 10 outer folds, the *diagquadratic* was selected seven times and the *linear* three times. However, the number of selected features varied a lot, ranging from 12 to 60 (*diagquadratic* median: 35 features, *linear*: 56). The median error (optimal validation) of those hyperparameters was 0.1788. The mean test error across outer folds was 0.2103 ± 0.0503. The best combination of hyperparameters for the model was a *diagquadratic* classifier with 17 features (lowest test error obtained: 0.1058).

Principal Component Analysis Variances of all the original features (Fig.13) had high values, while PCs (Fig.14) had a high variance only for the first ones. The covariances in original data were high (maximum: 10.0043) amongst a lot of features, while it was practically null (maximum: $3.5037e^{-13}$) for the data projected on PCs, confirming less correlation between PCs.

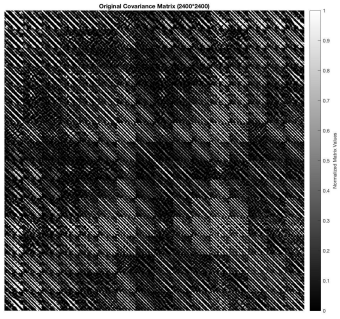

Figure 13: Representation of covariance matrices of original data. The white dots represent the variance on the diagonal elements, and the covariance on the off-diagonal elements. The whiter the dots, the higher the variance (or covariance).

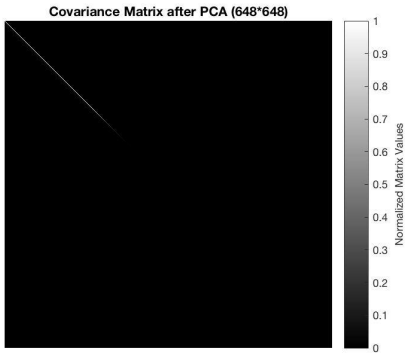

Figure 14: Representations of covariance matrices of the data projected on PCs. The white dots represent the variance on the diagonal elements, and the covariance on the off-diagonal elements. The whiter the dots, the higher the variance (or covariance).

The number of features projected on PCs selected in order to represent ∼90% of the total variance of the data was 44 features (Fig.15), explaining exactly 90.1966%.

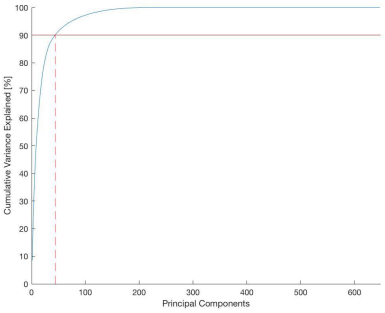

Figure 15: Percentage of cumulative variance explained in function of the number of PCs.

With the number of PCs as hyperparameter, the mean testing error across folds after CV was the smallest with 43 selected PCs (mean error: 0.4181).

Forward features selection After FFS, at each "outer" CV loop, the number of selected features were consistent : 6, 7, 7, 8, 8, 9, 9, 9, 13, 15 (median: 8-9), with a median of optimal validation error of 0.1419. The mean test error across outer folds was 0.2011 ± 0.0806.

Combining features engineering and selection with classifier types For the model combining PCA and *Fisher* ranking, the number of selected PCs varied across outer folds: 16, 24, 27, 29, 30, 34, 38, 40, 45, 53 (median: 32), with a median optimal validation error of 0.2214. The mean test error across outer folds was 0.2450 ± 0.0578. For the model combining FFS with classifier selection, the best classifier was systematically the *diagquadratic*. The number of features obtained for the outer folds was quite stable: 8, 11, 12, 12, 13, 13, 14, 15, 16, 22 (median: 13), with a median optimal validation error of 0.1090. The mean test error across outer folds was 0.1691 ± 0.0466.

Statistical significance The models tested for significance were respectively: *Fisher* feature ranking, *Fisher* feature ranking combined with classifier choice, FFS, PCA combined with *Fisher* feature ranking, and FFS combined with classifier choice. The test error distribution for each model seemed to be normally distributed (Kolmogorov-Smirnov test), as it can be seen on Fig.16. The test errors across the outer folds of each model all showed a statistical significance compared to the 0.5 random level: p-values found all below the alpha threshold of 5% ($3.5657e^{-6}$, $2.0524e^{-8}$, $9.4129e^{-7}$, $2.1093e^{-7}$, $3.2469e^{-9}$).

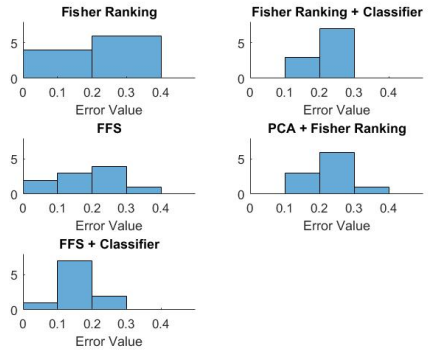

Figure 16: Test Error Distributions across Outer Folds for each model considered.

Final Classifier The model with the lowest mean test error was the one combining FFS and classifier choice (Fig 17). From this model, the *diagquadratic* classifier combined with the 12 features (hyperparameters) giving the lowest test error should be kept for subsequent model building.

	Mean Test Error across Outer Folds	Standard Deviation
Feature Selection (Fisher ranking)	0.2013	0.0944
Feature Selection (Fisher ranking) + Classifier Choice	0.2103	0.0503
Feature Selection (FFS)	0.2011	0.0806
Feature Engineering (PCA) + Feature Selection (Fisher ranking)	0.2450	0.0578
Feature Selection (FFS) + Classifier Choice	0.1691	0.0466

Figure 17: Table resuming the test errors across outer folds for all models evaluated.

4 Discussion

Statistical significance, Features thresholding and Generalization Features with similar distributions were not appropriate to distinguish erroneous movements from correct ones as no difference was observed in the different cases. On the contrary, good features could later be used to build a model of recognition, as they allowed a real distinction between classes. Those features were selected with the minimal p-value, and the threshold for separation of class A and B by choosing the minimal class error. It could be seen that samples were generally on the left of the threshold for Class A whereas samples on the other side corresponded mostly to Class B. The obtained error was still high, confirming the choice to investigate other classification methods. In assessing the generalization of this method, the optimal split between training and testing set was found to be 80% and 20% respectively. The behavior of the testing error with different splitting ratios could be explained by over-fitting: if a classifier is trained too much on a particular dataset, it becomes very specific to this dataset (low train error), but works less on a new set (high test error). The training class error, on the contrary, decreased with the increase of samples used for training, even though a stabilization was mostly observed. However, the shortcoming of features thresholding is that the classifier was built only thanks to one feature, thus accuracy could not be excellent.

LDA/QDA classifiers, Training/Testing errors, Number of parameters, Cross-validation and Confusion Matrix The low *quadratic* classifier error observed was due to the fact that the it uses a complex algorithm allowing to have a more specific classification (in opposite to a *linear* classifier where the separation is only done by the mean of a line), thus better fitting the data. On the other hand, a smaller error should have been obtained when the prior probabilities were set to 'empirical' as here 'uniform' prior probabilities did not represent the reality of the classes repartition. Indeed, in this project only 25% were erroneous movements, while in 'uniform' probabilities there was 50% chance that correct and erroneous movements happen. This observation allowed the elimination of the *quadratic* classifier. This could mean that *diagquadratic* classifier was the best fit, as it was the only one showing smaller error with an 'empirical' prior probability which, in addition, was the second lowest. The fact that the *quadratic* classifier could not be used to compute the subsequent errors is due to the fact that, depending on the data repartition, one or more classes may have singular covariance matrices in the case where data are redundant, making this classifier impossible to use as inversion of the matrix is needed. Correct movements of the cursor occurred in 75% of the cases and erroneous movements only in 25% of the cases. In order to give more weight to Class A to respect the real balance of events, one could imagine modifying the prior probabilities of the classifier and set them to 0.75 for Class A and 0.25 for Class B.

As expected, the training error was smaller than the testing error for every classifier. Indeed, when the classifier was trained on one set, it had a high performance when tested on the same set, as it was tuned for this dataset. On the contrary, when the classifier was tested on an *unseen* set, the performance was lower. The final interest was the testing error as it allowed to predict how the model responded to an unseen dataset. The classifier with the lowest test error was the *diagquadratic*, confirming the previous preference for it. By randomizing the dataset before separating it in two sets, the issue of equal distributions between sets was resolved. However, the improvement on the training error still did not improve the testing error. This could be due to the size of the dataset, since the obtained training and

testing sets contained only 324 samples each. Even though the samples were randomized, the number of samples was not necessarily sufficient to obtain an equivalent distribution for the two sets. The same reason could explain the variability of the performance observed. To conclude, it was not possible to trust the error results, and thus a CV was needed.

The more parameters a classifier has, the more precisely it builds the model and thus the less error there is. However, this also means that the model becomes more complex and the number of parameters should not exceed the number of samples. Compared to the number of samples (648), the number of parameters for the two linear classifiers and the *diagquadratic* one were quite reasonable, meaning they were robust for the dataset. However, the *quadratic* classifier was not robust enough as it had too many parameters compared to the number of samples (more than 5 times the number of samples). A classifier with parameters optimized on a reduced number of values is preferable because the complexity of the model is diminished. As the complexity of the covariance matrix was lowered in the case of diagonal classifiers (only non-zero elements on the diagonal), those classifiers were favored. Finally, between the *diaglinear* and *diagquadratic* models, the one with more parameters was elected, confirming the retention of the *diagquadratic* classifier.

Concerning the 10-fold CV results, the *diagquadratic* classifier gave the smallest CV error, as expected. Comparing 10-fold and *leave-one out* CVs, the error mean was lower with the latter CV as the training set was composed of a lot of samples (647), so the model was well tuned for the dataset. As the testing set in that case was only composed of one sample, there was no problem of over-fitting and training the classifier on a larger dataset could only be positive. On the contrary, the SD of the error was greater in the case of the *leave-one out* CV due to the error computed on one-sample testing sets, leading to values only of 0 or 1. Thus, the variation was extreme in-between each k-loop, influencing the SD to increase. Additionally, similar error for different partitions of the *leave-one out* CV could be explained by the fact that, because the folds were only composed of one sample, a re-partition just changed the order of the samples, which did not impact the calculation of the mean or the SD. On the contrary, the 10-fold CV differences observed were due to the randomness of the k-fold partition, but they were minimal because labels were used for the partition, guaranteeing a similar class distribution within folds. There was no advantages of having model performances varying because a stable and reliable model was wanted, meaning the less variation in SD, the more predictable the model. For this reason, and because the leave-one out CV is time consuming, we chose to continue the project with 10-folds CV.

Class error and confusion matrix are linked in the sense they return the performance of the model by giving errors, helping the assessment of the model's performance. As erroneous movements were initiating the ERPs of interest, the most important samples were the ones with an erroneous cursor movement. Thus, loosing the less information about them was essential. Here, there were more misclassified correct movements as erroneous than the other way around, which only brought impurities into the erroneous movements dataset without any information loss.

Cross-validation for hyperparameters optimization and nested cross-validation for performance estimation In *Pearson* method, important features could be missing, due to the fact that correlation could be close to 0 even if there was a close non-linear correspondence, thus lowering the number of selected features. This was confirmed by a lower number of selected features with the *Pearson* method compared to selection with *Fisher*. Moreover, the labels have a discrete nature whereas the values for a feature are continuous. Thus, using the *Pearson* method would not be reliable for features selection (it is indeed more appropriate for regression). This is why *Fisher* features selection was kept for the rest of the project.

Cherry-picking is the act of pointing to indi-

vidual data that seem to confirm a particular position, while ignoring a significant portion of related data that may contradict that position. The 10-fold CV for a normal random classifier with a theoretical error of 50% illustrated this. Each time CV was computed, a different error was found but the model with the minimal error was selected, even though in this case, no model should be better than another as they were randomly produced with a theoretical error of 0.5. Thus, one could conclude that testing different models and reporting only the best performance is a cherry-picking process. This is why a nested CV was needed, the outer CV enabling verification on the model choice.

With *Fisher* method, the median optimal validation error obtained with the nested CV was very close to the one found with the simple CV, confirming a good correspondence between the two implementations, both having a 10-folds (inner) partition. Concerning the nested CV, the errors computed on the train and validation sets showed nearby values, confirming coherent selection of the number of features. The test error had a greater value than the other errors and a much greater variation too, which was expected as it was computed on new data. This confirmed a good overall performance of the model, all errors staying around 0.2.

By selecting the classifier type on top of the number of features, the number of selected features did not stabilize and the performance did not improve. Furthermore, the number of selected features became much bigger, which increases the model's complexity. However, the most chosen classifier was the *diagquadratic* one, again confirming its retention. The *diaglinear* classifier was still the one fixed as it diminished the complexity of the model.

PCA, FFS, Combining the feature engineering and selection with classifier types As seen in the results, only the first PCs had a high variance, being the ones with the highest informative power. The most informative features were the ones having the greatest variance, enabling a certain differentiation between classes. As only few PCs had high variances, it was easier to choose which ones to implement for the model, while with the original features, the differentiation between classes could be computed by all features or a combination of several. For the original data, high covariances showed correlation between features and thus unnecessary redundancy. As the covariance of data projected on PCs was null (or extremely low), it indicated their independence (or very low dependence) from one another. Thus, PCA reduced the redundancy in the data. The optimal number of PCs obtained by CV was 43, as increasing the number of PCs after did not increase the performance of the model while increasing its complexity. This number was very close to the 44 PCs obtained previously from thresholding $\sim$90% of cumulative variance. For FFS, the results obtained showed there were less features needed (between 6 and 15) than with the *Fisher* features ranking (20 features selected). This could be explained by the fact that some features cochosen in the *Fisher* method may have been correlated, whereas FFS suppressed the ambiguity.

Conclusion The 5 estimated models were found to be significantly better than the random 2-class level (0.5). However, the first 4 had very similar overall mean test errors around 0.2, whereas the last one (FFS and classifier model) had a lower error, but it was a very time consuming program (several hours). To assess a balance between accuracy and time-consumption, the model could be repeated only taking into account one feature out of two, and the resulting error could be compared to the one obtained with all features. If the difference was not significant and still lower than 0.2, it would be the one selected. Otherwise, another more efficient model would be chosen. To increase the model reliability, an odd number of outer folds could also be considered, to obtain median error values corresponding to a real computed values (not the mean of two values). For example, 9 folds could be considered (less outer folds than inner ones is better to have enough data for our inner training set). We could also have tested our errors by changing the number of outer and inner folds, in order to find better ones.