Brain Fingerprinting: Identifying Individuals via learned brain graphs

ML4Science CS-433 in collaboration with MIPLab - Supervised by Hamid Behjat

Anabel SALAZAR DORADO, Lucille NIEDERHAUSER, Emilie MARCOU

***Abstract*—It has previously been shown that it is possible to identify individuals using brain activity signals such as functional Magnetic Resonance Imaging (fMRI) or magnetoencephalography (MEG), a process known as brain fingerprinting. This is generally done by computing statistical brain region dependencies to build functional connectivity (FC) matrices. In this work we explore a new approach of brain fingerprinting using MEG data that differs from previous methods in two ways. First, we use the brain signals to learn brain graphs instead of computing FC matrices. Secondly, we explore the use of machine learning classifiers to identify individuals instead of computing inter-subjects correlations. Our results show that this approach is a promising way of performing brain fingerprinting resulting in good accuracies and stable results across MEG frequency bands and graph learning parameters.**

I. Introduction

The brain is a highly complex structure and it can therefore be advantageous to find simpler ways to represent its physical and functional connections. Both type of dependencies can be modelled as a network where each node is a brain region and each edge a connection between them [1]. While structural connections represent the actual physical links between the regions [2], functional ones can be computed as the statistical relationships between the regions' signals [3] and represented by a functional connectivity (FC) matrix. This is a symmetrical matrix of size brain regions $\times$ brain regions where each element e_{ij} represents the strength of the statistical dependency between the region i and the region j of the brain. These matrices have been shown to be specific to each individual and thus it is possible to use them to identify people, a method known as brain fingerprinting [4][5][6].

We aim to build on Sareen et al's [6] work from 2021 where it has been demonstrated that subjects were identifiable using FC matrices built from MEG recordings. Indeed, when computing the correlations between a pool of FC matrices from many individuals built from a MEG session and FC matrices from the same individuals but another session, the highest correlation was generally between a subject and themselves. This methodology can thus be used to identify people. We want to see whether it is also possible to perform brain fingerprinting from MEG data by using an adjacency matrix instead of a FC matrix. Similarly to FC matrices, each element e_{ij} of an adjacency matrix represent the strength of the connection between the brain regions i and j. However, unlike FC matrices, these elements are not statistically computed but inferred by a graph learning algorithm. Moreover, we want to see how different parameters of this algorithm affect the brain fingerprinting capability of our methods. Furthermore, we aim to see whether a ML classifier could be used and how it compares to the aforementioned correlation method [6].

II. Data

The project was conducted using Human Connectome Project (HCP) data from resting-state MEG recordings from 84 patients. It was obtained on two separate runs named 'test' and 'retest'. The Destrieux cortical parcellation [7] was used, which defines 148 regions of interest. The data was pre-processed similarly to Sareen et al. [6]. This includes splitting of the time-reconstructed time series into different 12-seconds epochs. Depending on the subject, there were between 19 and 24 epochs. The signals were also band pass filtered into the five canonical frequency bands, namely, δ, θ, α, β, γ. From this, data from one patient, one epoch, one frequency band and one session was input into the graph learning algorithm to output an adjacency matrix. Additionally, we explored different hyper parameters of the graph learning algorithms like the type of regularization - log-regularization and l2-regularization - and the sparsity parameters - α for l2-regularization and β for the log-regularization. In the end we obtained one adjacency matrix per person, session, frequency band, epoch, regularization and value of the sparsity parameter (ranging from 0.05 to 1 with increments of 0.05). The data we extracted from the graph learning was presented in the form of a 4-dimensional matrix (W) with the first two dimensions being the adjacency matrix, the 3^{rd} dimension the sparsity parameter and the 4^{th} dimension the epochs. We had one W matrix for log-regularization (Wlog) and one for l2-regularization (Wl2) for each person, session and frequency band.

To extract features to feed into our machine learning algorithm, we took the adjacency matrix for a given frequency band, session and sparsity parameter and isolated the upper triangular part without the diagonal, flattened it and treated each value as a feature. Only the upper triangular matrix was taken since it is a symmetrical matrix, and we excluded the diagonal values given they are zeros (since they represent the functional connection between one brain region and itself). This was done for each person and epoch and concatenated together. Therefore, in the end we obtained a data set for each of the 20 sparsity parameter, 5 frequency bands, 2 sessions and 2

regularizations (400 data sets), each of with a number of rows equal to the sample size N (84) $\times$ number of epochs (19-24) and 10'878 features as columns. The 'test' sessions were used as the training sets for our models while the 'retest' sessions were used in order to test them and assess the accuracy of our predictions (i.e. as a our test set).

Another particularity of our data is the fact that in the W matrices we obtained from the graph learning, the last element in the epoch dimension of the matrix corresponds to the adjacency matrix obtained from when all epochs are combined to learn the graph. We treated those matrices separately as they are not the result of graph learning from one single epoch. We thus built 400 additional data sets as described above but this time containing only 84 samples. Indeed, since we combine all the epochs to learn the graph, we only obtain one adjacency matrix per person. The goal was to see whether learning a graph with all epochs combined would provide a better graph and could therefore yield better accuracies in our machine learning models.

III. Methods

A. Graph Learning

The graph-learning MATLAB script used for generating the adjacency matrices was provided to us. Multiple parameters define the graph learning algorithm. Firstly, the signal format can be either bandpass (bp), which is the bandpass filtered signal or envelope (env), the envelope of that signal. We only used the envelope. Additionally, it is possible to choose the measure of distance. We used the inner-vector product between pair of vectors.

Another parameter that heavily impacts the content of the adjacency matrices, is the sparsity parameter. We computed the sparsity of a matrix as the number of values below 5% of the max value of that specific graph divided by the total number of values. We investigated the impact of ranging this parameter from 0.05 to 1 across all bands and for the two regularizations. As can be seen intuitively on figure 1, increasing this parameter decreases the sparsity of the graph i.e., the matrix becomes more dense.

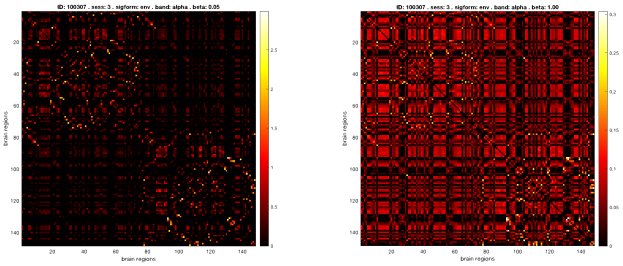

Fig. 1: Adjacency matrices obtained from the alpha band frequency and the log-penalty with a sparsity parameter set to 0.05 on the left, and 1 on the right. Here the matrix represents a single epoch.

We then plotted those values to be able to see the general trends. Each epoch was plotted as an individual line to assess whether some epochs showed atypical behaviours and should therefore be excluded from the data set. Such plot for alpha band and log/l2-regularization can be seen in figure 2. Across all bands and regularizations, we can see a decrease of sparsity with increased parameter value. The drop is more drastic in the log-regularization than in the l2 one.

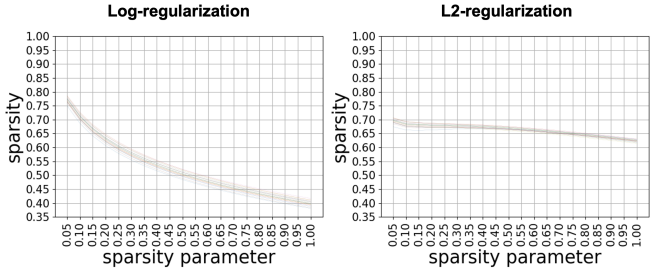

Fig. 2: Plots representing the sparsity with respect to the regularization parameter. Each of the 19 lines represents one epoch averaged over all subjects. The left-hand side plot is with log-, and the right-hand one with l2-regularization

B. Machine Learning Models

To see whether machine learning methods could be used to identify individuals from their adjacency matrices, we explored two different supervised classifiers. For each train and test data set (originating from one learning graph setting, i.e. one of the two regularizations and one specific sparsity parameter, but two different MEG session), we built two different multi-class classifier with 84 classes, the labels being the ID number of each subject. All our features were standardized before fitting the models to prevent some features taking over the others. For this we used the standardscaler function from the scikit-learn library [8]. Each train set was used to tune the hyper-parameters of our classifiers. To do so, we performed a grid search on different values for selected hyper-parameters (see sections III-B1 and III-B2) using 5-fold cross-validation to select the best combination of hyper-parameter. We did so independently for each of our 200 training sets built using single epochs as an input of the graph learning algorithm. Once the best hyper-parameters were selected, we trained our models on the entirety of each train set and then used each corresponding test set (same graph learning setting) to compute the accuracy of the 200 differently tuned models.

For the 200 training sets built from the adjacency matrices learned using all epochs combined, we used a different method. We did not perform a grid-search, but trained our models using hyper-parameters that showed to work well for the data sets with the same regularization and sparsity parameters but built using adjacency matrices learned using all epochs separately. This thus means that the performance of these models can still certainly be improved since the range of possible hyper-parameters was not explored.

1) Support Vector Machine: The first model we used to classify our subjects was a Support Vector Machine (SVM)

model. To implement it we used the SVC function from the scikit-learn library [8]. The different hyper-parameters we explored during our grid-search are listed below with a short description. For the unlisted hyper-parameters, we simply set them to their default values in the scikit-learn library [8].

- *C*: Regulates the strength of the L2 penalty and is inversely proportional to the strength of the regularization [8]. Values explored: 0.1, 1, 10 and 100.
- *kernel*: The kernel that will be used in the algorithm [8]. Kernels explored: *rbf*, *sigmoid* or *polynomial*.
- *gamma*: Gamma defines the curvature of the decision boundary when the kernel is *rbf*, *sigmoid* or *poly* [8]. Values explored: 10, 1, 0.1, 0.01, 0.001 and *scale*. Where *scale* is (number of features $\times$ variance of the feature matrix)$^{-1}$

We realised that our best models across all graph learning settings were always using a regularization coefficient of 1 or above, meaning that our models need to have a strong regularization, which is not surprising since we have significantly more features than samples. Therefore, the model without regularization will easily over-fit the training set and thus give a bad performance on the validation set. The other hyper-parameters varied greatly depending on the train set used.
To improve the performance of this model, we used some pre-processing techniques. Given the amount of features we had, we tried reducing that number. We first wanted to perform a Principal Component Analysis (PCA) in order to identify the most relevant features. However, the variance of our features ranged from 10^{-8} to 10^{-2} and was overall too low to conduct such an analysis. Given that high variance features tend to better explain the data, we started removing columns with low variance and observed how that impacted the performance of the SVM. By setting a threshold variance, and removing all features with variance below that, we observed that in some settings, even by reducing the number of features down to 200 (50x less than originally), we could still obtain a good accuracy. In most cases removing a lot of features even improved our accuracy. In the end, we performed 5-fold cross-validation on each of our training sets to see which number of feature lead to the best performance. We ended up with training sets with a number of features between 20 and 2300 depending on the graph learning hyper-parameters. We used these number of features to compute the SVM accuracies presented in table I and in the annex.

2) Random Forest: For our second classifier, we used the Random Forest Classifier also from the scikit-learn library [8]. Similarly to the method used for our SVM model, we explored different hyper-parameters listed below. For this model as well, hyper-parameters that are not specified below were left as their default values in the scikit-learn library [8].

- n estimators: The number of trees built in our forest, before averaging them [8]. Values explored : 100, 500 and 1000
- max depth: The maximum depth of the trees built [8]. Values explored: 50, 100, 1000 or no maximum depth. In this last case, the tree is built until each leaf contains less than *min samples split* samples [8].
- min samples split: The minimum number of samples required to keep building the tree, i.e. to split a node [8]. Values explored: 2, 5 and 10.
- min samples leaf: The minimum number of samples that are allowed at a leaf node [8]. In other words, a split resulting in a node with a number of samples smaller than this parameter will not be allowed. Values explored: 1, 2 and 4.

Our best models always had a maximum depth of 50 or 100. Similarly to using low C coefficient in SVM, higher trees depths can lead to an over-fit of the training set and thus do not perform well on the validation sets.

C. Correlations

To benchmark our results against a method more similar to what has been done before in the field of brain fingerprinting, and explore whether using machine learning in this context is advantageous, we used a statistical approach to classify our subjects based on Sareen et al's work [6]. To each person in one of our test sets we assigned the ID of the subject in the corresponding training set with which it had the highest Pearson's coefficient of correlation. We did this with all our 400 train set-test set pairs. In the same way it is traditionally done with machine learning classifier, we computed the accuracy of this method as the percentage of people correctly classified in the test sets.

IV. Results

Our best accuracies and their corresponding hyper-parameters are summarised in table I. Our highest accuracy is 97.6% using the SVM classifier.

Model	Type of data set	Hyper-parameters	Accuracy
SVM	Single epochs	C = 10 gamma = scale kernel = sigmoid	0.971
SVM	All epochs combined	C = 1 gamma = scale kernel = sigmoid	0.976
Random Forest	Single epochs	n estimators = 500 max depth = 50 min samples split = 2 min samples leaf = 1	0.962
Random Forest	All epochs combined	n estimators = 1000 max depth = 50 min samples split = 2 min samples leaf = 1	0.917

TABLE I: Summary of our best models and their corresponding hyper-parameters. *Type of data set* refers to whether the models were trained and tested on the data sets built using the adjacency matrices learned from all epochs combined from one subject or from single epochs.

As we can see from figure 3, SVM after having been optimised with feature removal (as seen in section III-B1), reaches

higher mean accuracies than Random Forest ($\mu_{SVM} = 0.908$, $\mu_{RF} = 0.89$). However, we also observe that the variance is significantly higher for SVM than for Random Forest ($\sigma^2_{SVM} = 0.001699$, $\sigma^2_{RF} = 0.000349$). Moreover, we observe significantly more outliers towards smaller accuracies in the SVM model. We can thus infer that, although it reaches lower mean accuracies, Random Forest is overall more stable across the different frequency bands and regularizations. Therefore, for the rest of our analysis, we will focus on this method and compare it to the correlation method. The results we obtained for SVM are nevertheless presented in the annex.

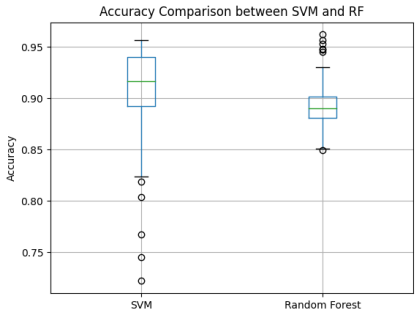

Fig. 3: Accuracy comparison for our two machine learning algorithms across frequency bands and regularization

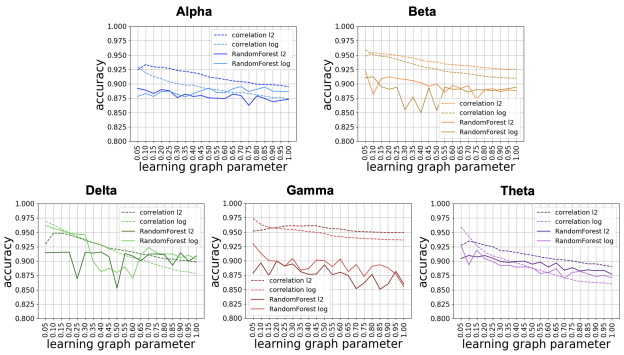

Fig. 4: Accuracy plots for the Random Forest classifier and the correlation method for both regularization and for all frequency bands.

In figure 4, we compare for each frequency band the correlation method (dashed lines) with Random Forest (solid lines) for both regularizations. The first observation to make is that, as expected, our model has a similar performance across all band, regularizations and sparsity parameters. We also observe that for two frequency bands, beta and gamma, the correlation method clearly outperforms our model. This is not due to the fact that the Random Forest model performs worse in these bands but simply the correlation models performs particularly well in them. We can therefore conclude that the correlation method works especially well in higher frequency bands. However, in this method, the accuracy tends to decrease more strongly with increasing parameter values than the Random Forest model. The correlation method therefore seems to be more impacted by change in the sparsity of the learned graph leading to better or comparable performance of both models at higher sparsity parameters. We can therefore conclude that machine learning classifiers can offer the advantage of having more robust predictions, regardless of the frequency band or the sparsity of the adjacency matrix.

The same pipeline was applied to the data sets built from graphs learned on all epochs and results are presented in the annex. More variable accuracies were obtained which is not surprising as no grid search was performed to find optimal hyper-parameters as discussed in section III-B

V. Discussion

Previous method of brain fingerprinting using MEG data and functional connectivity (FC) matrices, gave accuracies ranging from 52.9% to 98.2% depending on how the FC matrices were built and which frequency bands were used as input data [6]. Even though we do not reach an accuracy as high as 98.2% with any of the methods presented here, they all do show very stable results across all frequency bands and sparsities of the adjacency matrices explored. This indicates that brain fingerprinting using learned brain graphs instead of FC matrices is a promising method of identifying individuals that should be further explored. For instance, one could study how using different measures of distances in the graph learning algorithm would affects brain fingerprinting. The current machine learning classifiers could also be further improved by exploring more values of the SVM and Random Forest hyper-parameters and better pre-processing of our data sets. Moreover, it would be interesting to see whether deep learning methods would be able to give higher prediction accuracies. Finally, since the features in our data sets correspond to connections between brain regions, ranking the features by order of importance to correctly classify individuals would give us information on which brain regions have the highest fingerprinting capabilities and thus gain insights on the connections that differ from individual to individual.

VI. Conclusion

Brain fingerprinting via learned brain graphs using MEG data gives relatively high accuracy using either a statistical approach (correlations) or a machine learning approach (Random Forest classifier), both resulting in similar classification performances (from 85% to 97% accuracy). A SVM classifier gives more variable results, but is also able to perform fingerprinting. Knowing that ML classifiers are an alternative to correlations is an important finding as it can reduce the time and computational resources needed to perform brain fingerprinting. Moreover, we showed that the ability of identifying individuals does not depend on the frequency bands extracted from the MEG signals nor the sparsity of the adjacency matrix inferred by the graph learning algorithm. Our work thus suggests that learned brain graphs combined with a ML classifier could be an effective and accurate way to perform brain fingerprinting.

REFERENCES

[1] Bullmore, E. & Sporns, O. Complex brain networks: graph theoretical analysis of structural and functional systems. *Nature reviews neuroscience* **10**, 186–198 (2009).

[2] Hagmann, P. From diffusion mri to brain connectomics. Tech. Rep., EPFL (2005).

[3] Friston, K. J. Functional and effective connectivity in neuroimaging: a synthesis. *Human brain mapping* **2**, 56–78 (1994).

[4] Finn, E. S. *et al.* Functional connectome fingerprinting: identifying individuals using patterns of brain connectivity. *Nature neuroscience* **18**, 1664–1671 (2015).

[5] Amico, E. & Goñi, J. The quest for identifiability in human functional connectomes. *Scientific reports* **8**, 1–14 (2018).

[6] Sareen, E. *et al.* Exploring MEG brain fingerprints: Evaluation, pitfalls, and interpretations. *NeuroImage* **240**, 118331 (2021). URL https://www.sciencedirect.com/science/article/pii/S1053811921006078.

[7] Destrieux, C., Fischl, B., Dale, A. & Halgren, E. Automatic parcellation of human cortical gyri and sulci using standard anatomical nomenclature. *Neuroimage* **53**, 1–15 (2010).

[8] Pedregosa, F. *et al.* Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research* **12**, 2825–2830 (2011).

Annex

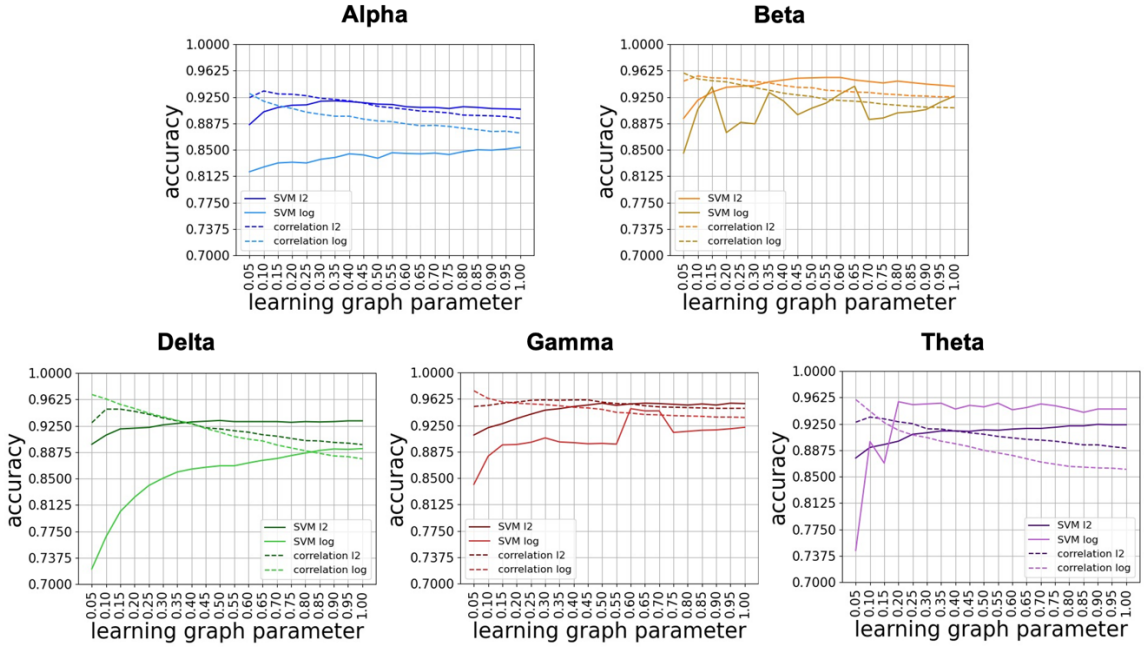

Figure 1: Result Plots for SVM Algorithm compared with benchmark for all frequencies

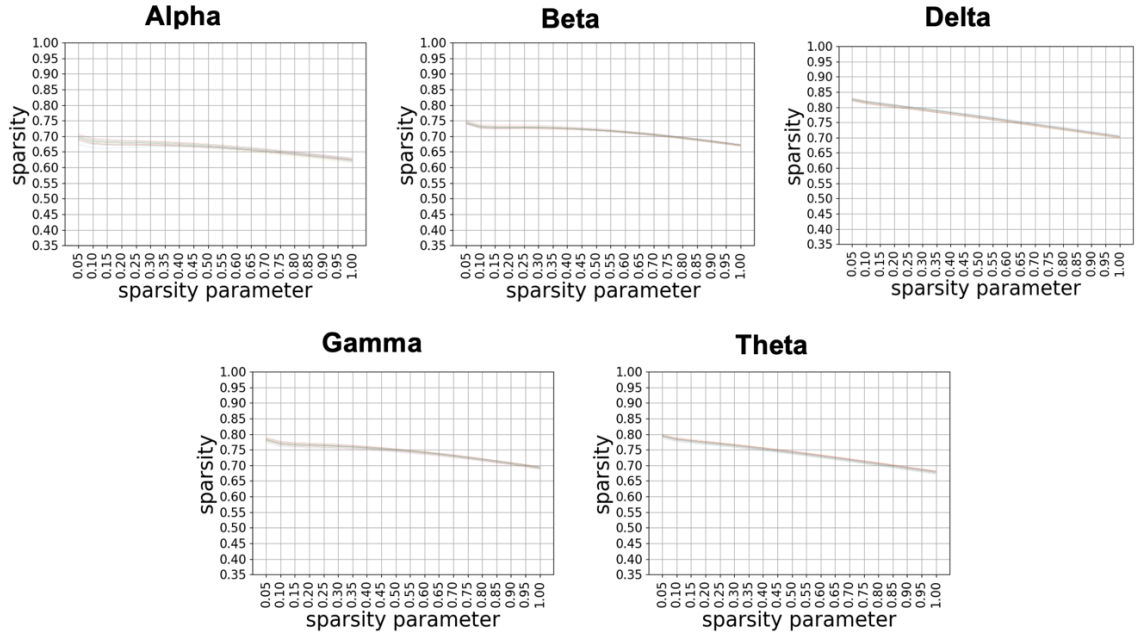

Figure 2: Sparsity for all epochs for the l2 regularization. Each line represents an epoch. The x-axis has increasing values of the sparsity parameter

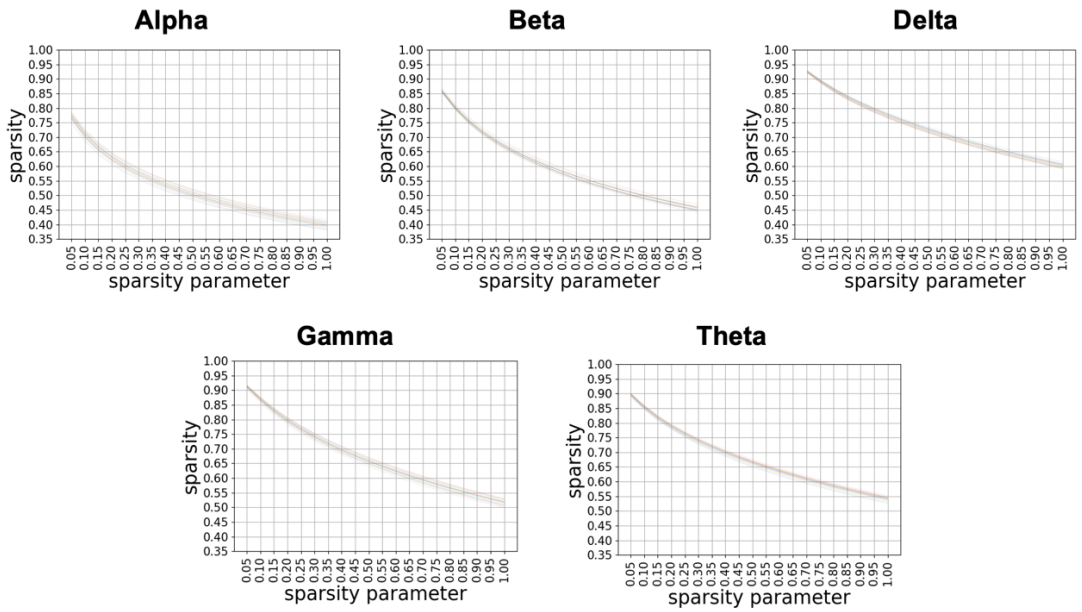

Figure 3: Sparsity for all epochs for the log regularization. Each line represents an epoch. The x-axis has increasing values of the sparsity parameter

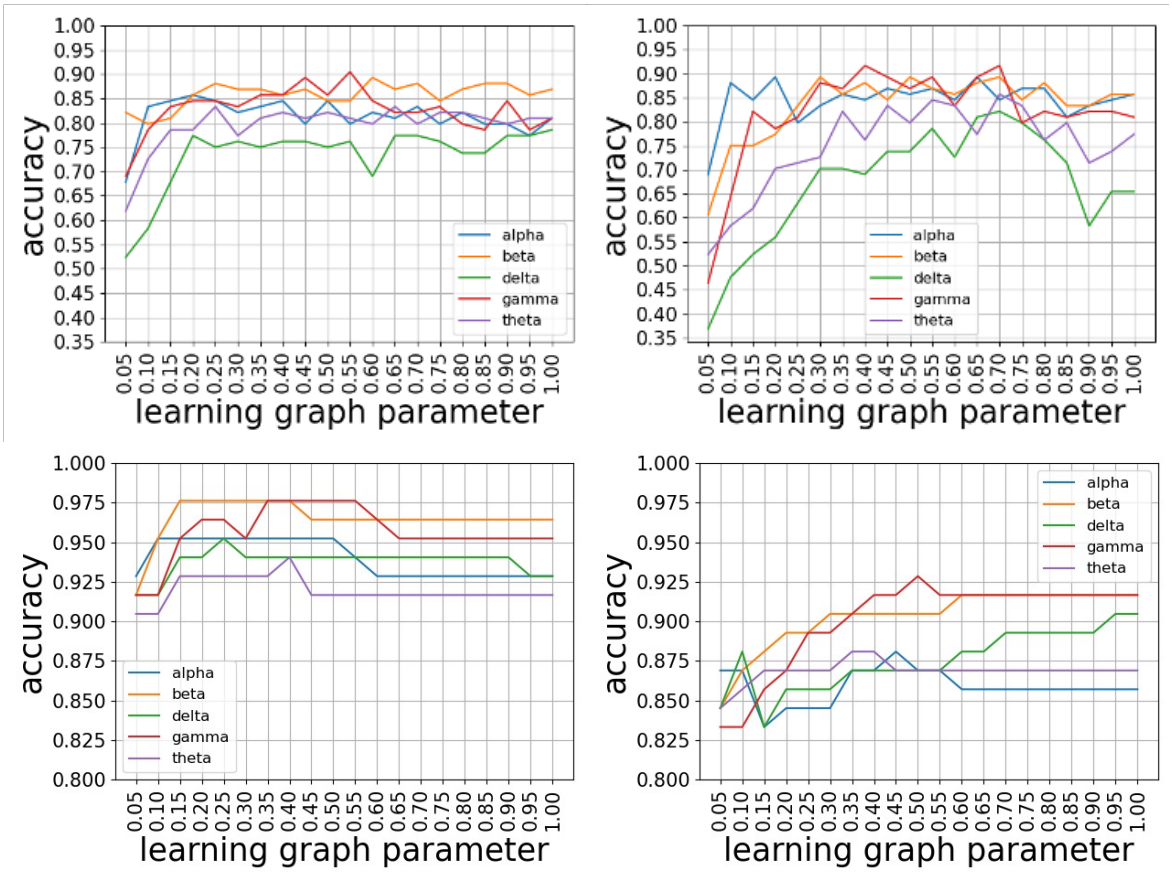

Figure 4: Results for the RF and SVM algorithm when considering the graph learned on all epochs. The top two graphs are from RF and the bottom two with SVM. The left column represents l2- and the right column the log-regularization