Classification of Brain Networks using Dirichlet Distribution of Graph Spectra

Anna Tkachev
National Research University Higher School of Economics, Moscow, Russia
annatkachev42@gmail.com

Yulia Dodonova
National Research University Higher School of Economics, Moscow, Russia
ya.dodonova@mail.ru

Abstract

In this work, graph spectra of the normalized graph Laplacians of brain networks (connectomes) are used for solving the task of classifying autism spectrum disorder against typical development. We find the most informative group of eigenvalues by introducing a window and sliding it through all possible positions. We next assume that these values are sampled from a Dirichlet distribution and build a linear model with a single feature that is based on estimation of a Dirichlet parameter. The proposed classifier outperforms the baseline in terms of both mean ROC AUC value (0.74) and stability of ROC AUC values to the variations in the data. Classifiers that implemented a similar approach but used geometric distances instead of statistical methods showed worse performance. This implies that the Dirichlet distribution might be a useful tool for the analysis of normalized Laplacian spectra when solving tasks of classifying brain networks.

Keywords. connectomes, graph spectra, graph Laplacian, Dirichlet distribution

1. Introduction

One way of exploring the human brain is by representing it as a network of macroscale neural connections, and then studying properties of this network. These networks (or graphs) are called connectomes. Connectomes are constructed from neuroimaging data so that nodes represent brain regions and edge weights show some measure of structural or functional connection between the regions.

It can be hypothesized that these networks differ in normal and pathological brains and hence network representation of human brains can help to classify normal and pathological development. However, to be able to run machine learning algorithms one need to construct graph-based features that capture most informative structural properties of the respective networks.

We propose that spectra of the normalized graph Laplacians of brain networks are particularly useful for a task of classifying connectomes. In our previous works [1],[2], we mainly considered the shape of the distribution of the eigenvalues, computed pairwise distances between the spectral distributions and next run a support vector machines (SVM) classifier with the kernels based on these distances. In [1], we explored behavior of the kernels that were based on information-theory distances: the Kullback-Leibler divergence and the Jensen-Shannon distance. However, these kernels required density reconstruction, and the outcomes of classification were extremely sensitive to the parameter used to reconstruct density (the number of bins). For a task of classification of autism spectrum disorder versus typical development, the area under the receiver operating characteristic curve (ROC AUC) was about 0.65 on average, although with some arbitrary numbers of bins it unsystematically peaked higher. In [2], we aimed to overcome this problem by using the earth mover's distance (EMD) as a measure of dissimilarity between spectral distributions. For the same task of classifying autism spectrum disorder against typical development, we obtained classification quality of 0.71 (in terms of ROC AUC) using the SVM with the EMD-based kernel.

Unlike our previous works, in this paper we are not considering the shape of the spectral distributions of brain networks. Instead, we are considering each vector of eigenvalues as a vector of "proportions" and using appropriate measures of similarity.

In what follows we give the motivation and details of the proposed approach and show how it works for the task of classifying autism spectrum disorder against typical development. We are using a set of precomputed structural connectomes, the same dataset as in our previous works. We show that our proposed approach works better for the given classification task, implying that this model might allow to better grasp the significance of graph spectra in representing properties of brain networks.

2. Graph Spectra

Working with graphs in a machine learning framework can be tricky. One way to produce feature vectors to be used in machine learning algorithms is to consider a spectrum of some matrix corresponding to the graph. In our study we are working with the spectrum of the normalized graph Laplacian, which is used in graph theory to derive many useful graph properties.

Given an undirected weighted graph with n nodes, we define the adjacency matrix A as the $n \times n$ matrix with entries a_{ij}, where a_{ij} is the weight between the respective nodes (in this study, we deal with weighted graphs where the weights are not binary).

We define D to be a diagonal matrix of weighted node degrees:

$$d_i = \sum_j a_{ij}. \tag{1}$$

The normalized graph Laplacian is given by:

$$\mathscr{L} = D^{-1/2}(D-A)D^{-1/2}. \tag{2}$$

Let $\lambda_{i,0},...\lambda_{i,(n-1)}$ be the eigenvalues of the normalized Laplacian of the i-th connectome, sorted in ascending order:

$$\lambda_{i,0} \leq ... \leq \lambda_{i,(n-1)}$$

The spectrum of the normalized graph Laplacian can provide us with useful properties of the graph. For example, in [3] the authors show that specific patterns in the graph result in specific eigenvalues. For a general theory on the normalized graph Laplacian and its spectrum, we refer to [4] or [5].

In this work we will only talk briefly of some particular properties. The first property is that the eigenvalues of the normalized Laplacian are always in the range from 0 to 2:

$$0 \leq \lambda_{i,j} \leq 2$$

Second, for the normalized Laplacian spectrum of any network the following holds:

$$\lambda_{i,0} = 0 \tag{3}$$

In context of this study, this means that there is no variability between networks in the value $\lambda_{i,0}$. We eliminate it from all the analyses. In what follows, we denote the vector of eigenvalues:

$$\Lambda_i = (\lambda_{i,1}, ...\lambda_{i,(n-1)}) \tag{4}$$

The third property, which is most important for this work, is that the sum of the eigenvalues always equals n:

$$\sum_{j=0}^{n-1} \lambda_{ij} = n \tag{5}$$

This property motivates the method used in this work. In our previous works ([1] [2]), we considered the individual components λ_{ij} of a given sample i to be instances of a univariate random variable, and computed similarity between two vectors Λ_i, Λ_s using distances between probability measures. In this work, we assume the vector $\frac{\Lambda_i}{n}$ to be sampled from the r-variate Dirichlet distribution ($r = n-1$) and compute similarities between vectors of eigenvalues based on this assumption.

3. Dirichlet Distribution

The Dirichlet distribution is one of the most known multivariate distribution that models random vectors with non-negative components that sum up to a constant.

Let $(p_1,..p_r)$ denote a random vector, $p_i > 0$, $\sum_j p_j$=1. Then the probability density of the Dirichlet distribution with vector of parameters $(\alpha_1,..\alpha_r)$ is given by:

$$\frac{\Gamma(\sum_{j=1}^r \alpha_j)}{\prod_{j=1}^r \Gamma(\alpha_j)} \prod_{j=1}^r p_j^{\alpha_j - 1}, \tag{6}$$

where Γ is the gamma function.

An intuitive way to interpret a vector sampled from the Dirichlet distribution with the vector-parameter $(\alpha_1,..\alpha_r)$ is to see it as a collection of proportions in which one cuts a segment of length 1 into r parts. The lengths of each of these r parts can vary slightly around a given mean, but not too much. The vector of these mean lengths will be the vector $(\frac{\alpha_1}{\sum_j \alpha_j},.. \frac{\alpha_r}{\sum_j \alpha_j})$. This gives us the intuitive interpretation of the parameter of the Dirichlet distribution. More on the Dirichlet distribution can be read, for example, in [6].

A standard way of finding parameter estimates for probability distributions is to minimize the maximum likelihood function, and these estimates are called maximum likelihood estimates (MLE). For the Dirichlet distribution, there is no formula for the MLE, but the paper [7] provides iterative schemes for obtaining the parameter. We refer you to this paper if you want more details about the MLE and likelihood function of the Dirichlet distribution. We used the python library *dirichlet* [8] that is based on this paper, and used the *mle* and *loglikelihood* functions from this library.

4. Data and Preprocessing

In this work, we use a publicly available dataset [9], [10] that includes precomputed DTI-based matrices of structural connectomes of 51 high-functioning autism spectrum disorder (ASD) subjects (6 females) and 43 typically developing (TD) subjects (7 females). Average age (age standard deviation) is 13.0 (2.8) years for ASD group and 13.1 (2.4) years for TD group.

Nodes for connectomes are defined using a parcellation scheme that is based on a large meta-analysis of fMRI studies combined with whole-brain functional connectivity mapping. This approach produces 264 equal-size brain regions and thus 264×264 connectivity matrices.

Network edges result from brain deterministic tractography. It is performed on voxelwise fractional anisotropy values using the fiber assignment by continuous tracking (FACT) algorithm. Edge weights in the original matrices are proportional to the number of streamlines detected by the algorithm. We additionally weight the edges by the squared distances between the respective regions:

$$a_{ij} = \frac{a_{ij}^{raw}}{l_{ij}}, \tag{7}$$

where l_{ij} is the Euclidean distance between centers of the regions i and j computed based on the MNI coordinates of region centers provided by the authors of the dataset.

The authors of the dataset only report the results of group comparison based on global graph metrics [10]. The best available classification baseline is ROC AUC 0.77 obtained by [11]. The authors trained linear SVM on node degrees computed from matrices weighted by squared Euclidean distances between regions and normalized by the geometric mean of the adjacent node degrees. That was the same weighting scheme (7) as we use in this study. However, we do not apply that normalization procedure because it partly replicates the computation of the normalized Laplacians.

5. Methods

To compare the performance of different classifiers, we use the area under the ROC-curve (ROC AUC) metric. Because the sample size is quite small, we compute the out-of-fold mean for the ROC AUC (and not the mean ROC AUC inside of the cross-validation). To do this, we train our model inside the cross-validation, each time yielding a vector of predicted probabilities for test objects, and then join these vectors into one, which is of the same length as the target vector. We then use this vector to compute the ROC AUC. Doing this procedure for different partitions of the data into folds, we calculate the mean value and standard deviation of the ROC AUC. In this paper, we used 10 different random partitions of the data into 5 folds.

5.1. Eigenvalues as Features

To produce a baseline, we use logistic regression classifier with l2 regularization with the eigenvalue vectors (4) as the features. Note that the eigenvalues are first sorted in ascending order.

5.2. Dirichlet-Based Classifier

In this work, we propose the following approach. Based on the vectors of eigenvalues, we construct a single feature using the Dirichlet distribution. Having this single feature, we then build a linear classifier, which is in this case simply a point on the real line that separates the two classes.

To be precise, this is the procedure. Assuming the data to have a Dirichlet distribution, we estimate its parameter vector from the data (using only the train data). Afterwards, we calculate the loglikelihood that each (test) sample is generated by the Dirichlet distribution with this parameter. In other words, let Λ_i be the vector of eigenvalues as defined in (4), M and N be the sets of train and test indexes of the objects, respectively. Then for each sample $i \in N$ we get a value:

$$f_i = \mathsf{L}\ (\ \alpha^{mle}\{\Lambda_k | k \in M\}\ \mid\ \Lambda_i), \tag{8}$$

where α^{mle} is the maximum likelihood estimate of the Dirichlet distribution parameter from the train sample, and L is the loglikelihood of this parameter given a test object i. The values of α^{mle} and L are computed using the *mle* and *loglikelihood* functions from the python library [8]. We next use the vector $F = \{f_i\}$ as a single feature vector.

As a side note, we want to mark that the initial idea was to compute the similarity of test objects to the average distribution of the first and second classes separately. In practice, however, the parameters estimated from the two classes separately turned out to be almost identical, which of course gave us two very highly correlated features.

5.3. Adding a Sliding Window

The method from the previous section did not give satisfactory results. We reasoned that $(n-1)$ dimensions (which is 263 in this study) might be too high, and that most eigenvalues might actually be unnecessary noise and not contribute to the classification performance. To tackle this problem, we introduce a sliding window that leaves intact μ central eigenvalues $(\mu < n)$ and sums up together the eigenvalues on the left and on the right of the window, producing two numbers in addition to the central μ values. Because the sum of the eigenvalues is still n, we can redo the procedure from the previous section, but with vectors with $(\mu+2)$ elements instead of $(n-1)$. We then optimize the performance of the classifiers over all possible positions of the window.

In other words, let

$$\Omega_i = (\sum_{s=1}^{\tau} \lambda_{i,s},\ \lambda_{i,(\tau+1)},\ \lambda_{i,(\tau+2)},\ \ldots,\ \lambda_{i,(\tau+\mu)},\ \sum_{s=\tau+\mu+1}^{n-1} \lambda_{i,s}) \tag{9}$$

be the vector of length $(\mu+2)$ and i the index of the sample, then for the i-th sample we get one feature:

$$\tilde{f}_i = \mathsf{L}\ (\ \alpha^{mle}\{\Omega_k | k \in M\}\ \mid\ \Omega_i), \quad i \in N, \tag{10}$$

where α^{mle}, L, M, N are defined as in (8). We next use the vector $\tilde{F} = \{\tilde{f}_i\}$ as a single feature vector. We optimize the performance of the linear classifier over the value τ, $\tau \in \{1,2,...,n-\mu-2\}$, which defines the position of the sliding window. Throughout this study, we only work the value of μ fixed at 20.

5.4. Alternative Approaches

We next question whether the results similar to those obtained using Dirichlet-based model can be obtained without actually making the assumption that the eigenvalues of the normalized Laplacian are sampled from the multivariate Dirichlet distribution. We evaluate two ideas.

First, we add a sliding window before running a baseline logistic regression classifier. In other words, we now run logistic regression classifier with l2 regularization using the feature vectors Ω defined by (9); thus, we now deal with $(\mu+2)$ features $(\mu = 20)$. We thus test the idea that adding a sliding window is the most important step that improves the performance of any classifier on the vectors of eigenvalues.

Second, we evaluate the idea that using a particular model, i.e. a Dirichlet-based model, might not be crucial for the proposed pipeline. Indeed, by constructing a feature $\tilde{F}$ (10) we estimate how close our test objects are to an estimate of some "average" obtained from a train sample. Classification based on this feature means that we expect objects from one class to be close to this "average" estimate, while objects from the other class to differ from it. In (10), the value of this "average" is estimated as a parameter of the Dirichlet distribution, and the closeness is defined in terms of probability of a particular test object to come from the Dirichlet distribution with the estimated parameter. We next simplify the procedure and test whether other proxies of the distance from some "average" estimate can also be useful for the purpose of classification.

To test this idea, we first estimate the mean vector of Ω as the most obvious proxy of the parameter of the Dirichlet distribution (this step is again based on the train data) and then calculate the distance from each (test) sample to the mean vector:

$$g_i = \rho(\Omega_i, \frac{1}{|M|}\sum_{k\in M}\Omega_k), i \in N, \tag{11}$$

where M and N are defined as in (8), Ω_i is defined by (9), $\mu = 20$. We use three different distance measures ρ: euclidean, cosine and chi-squared distances.

sect.	Classifiers	AUC mean	AUC std
5.1	Eigenvalues, logistic regression	0.643	0.046
5.2	Dirichlet likelihood, all eigenvalues	0.666	0.004
5.3	Dirichlet likelihood, sliding window	0.739	0.003
5.4	Eigenvalues, logistic regression, sliding window	0.671	0.024
5.4	Euclidean distance, sliding window	0.676	0.012
5.4	Cosine distance, sliding window	0.664	0.016
5.4	Chi-squared distance, sliding window	0.685	0.010

Table 1. Best results for classifiers from section 5

Euclidean distance is given by:

$$\rho_{Euclidean}(\Omega_i, \Omega_j) = \sqrt{\sum_{k=1}^{\mu+2}(\omega_{ik} - \omega jk)^2} \tag{12}$$

Cosine distance is given by:

$$\rho_{cosine}(\Omega_i, \Omega_j) = \frac{\sum\limits_{k=1}^{\mu+2}\omega_{ik} * \omega_{jk}}{\sqrt{\sum\limits_{k=1}^{\mu+2}\omega_{ik}^2}\sqrt{\sum\limits_{k=1}^{\mu+2}\omega_{jk}^2}} \tag{13}$$

Chi-squared distance is given by:

$$\rho_{chi-squared}(\Omega_i, \Omega_j) = \sum_{k=1}^{\mu+2}\frac{(\omega_{ik} - \omega_{jk})^2}{\omega_{jk}} \tag{14}$$

We next construct linear classifiers based on a single feature $G = \{g_i\}$ (11) that use (12), (13), or (14) as a distance measure, and compare their performance with that of the proposed Dirichlet-based classifier.

5.5. Tools

We used Python and IPython notebook platform, specifically NumPy, SciPy, pandas, matplotlib, scikit-learn and dirichlet [8] libraries.

6. Results and Discussion

In this section, we discuss the performance of the classifiers in the same order as they are presented in Section 5. Table 1 summarizes the results of all experiments.

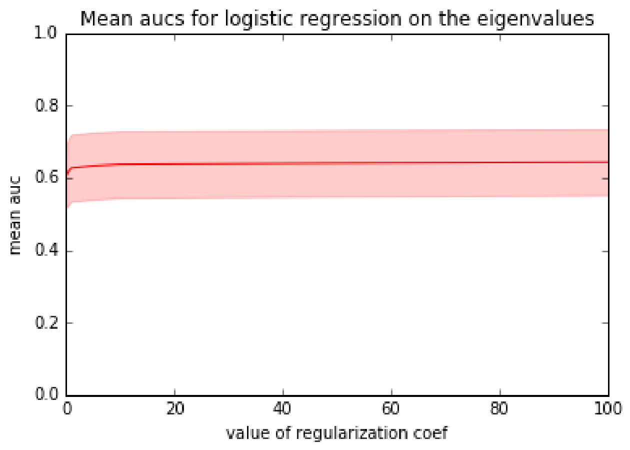

Figure 1. Logistic regression on all eigenvalues, ROC AUC mean ($\pm$ 2 standard deviations)

6.1. Logistic Regression on All Eigenvalues

Figure 1 shows the performance of logistic regression with l2 regularization on all of the eigenvalues (i.e., on the vectors of features defined by (4)). We plot the mean ROC AUC value over different 5-fold cross-validation splits against the values of the regularization parameter. You can see that regularization doesn't increase performance of this classifier. The highest ROC AUC value obtained with this classifier is 0.643.

6.2. Dirichlet-Based Linear Classification

We next evaluate the performance of the linear classifier that is based on a single feature $F = \{f_i\}$, with f_i defined by (8). Note that with this single feature, our classifier is simply a point on the real line that separates the two classes. The classifier has no parameters to be optimized. As shown in Table 1, this classifier produces ROC AUC value 0.666, which does not outperform the baseline. However, the standard deviation of ROC AUC values across different splits of the data is much lower than that of the baseline classifier.

6.3. Dirichlet-Based Classification with a Sliding Window

Next, we modify our Dirichlet-based classifier by adding a sliding window. We now work with $\tilde{F} = \{\tilde{f}_i\}$ for which the elements are defined by (10). Figure 2 shows the performance of this modified classifier. The x-axis corresponds to the index of the eigenvalue that stands in the middle of the window, i.e. the value $(\tau + 11)$. The blue vertical band shows the optimal position of the center of the sliding window (we will draw this band in blue on other plots too for reference).

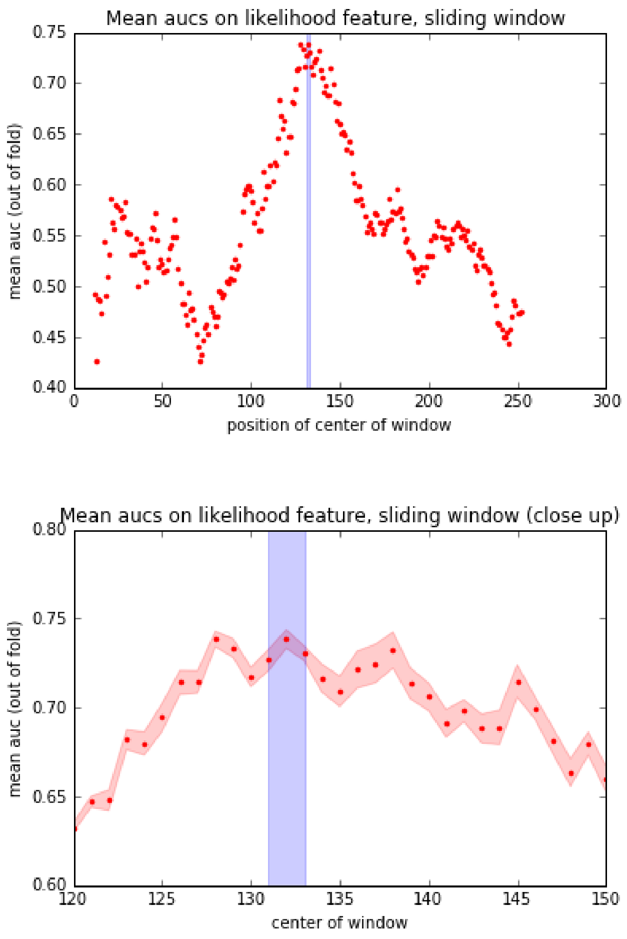

Figure 2. Dirichlet-based model with a sliding window, mean ROC AUC ($\pm$ standard deviation in the bottom)

Interestingly, the optimal position of the center of the window is around the value 132, which also happens to be the center of the original vector of eigenvalues. In other words, central eigenvalues are the most informative for our classification task.

The ROC AUC value of the Dirichlet-based classifier obtained with the optimal position of a sliding window is 0.739, which clearly outperforms the baseline.

Importantly, the standard deviation of ROC AUC values across different cross-validation splits of the data stays very low throughout all of the positions of the window. This happens because the parameter vector can be estimated on any given subset of reasonable size, and the results from different subsets are very close in terms of the loglikelihood function. This means that the classification quality does not depend that much on the partition of the data into folds, only on the sample itself – thus the low standard deviation of ROC AUC.

As mentioned above, our feature $\tilde{F}$ represents the

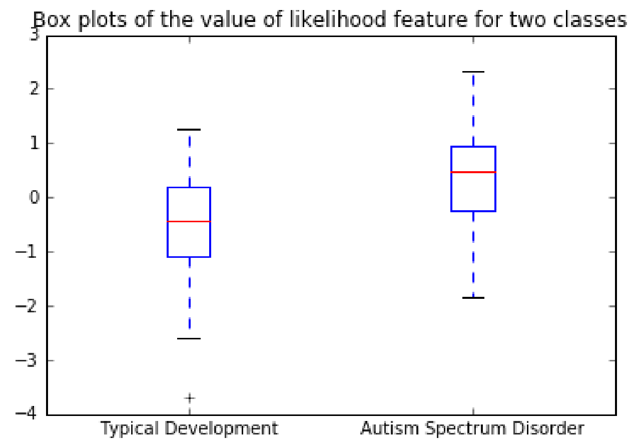

Figure 3. Values of the Dirichlet-based feature $\tilde{F}$ in two classes

probability of the sample to come from the assumed Dirichlet distribution with the estimated "average" parameter. High classification quality obtained based on this single feature means that objects from one class are systematically more likely to come from this "average" distributions than objects from the other class. Figure 3 illustrates this idea: it shows the boxplot of the values of $\tilde{F}$ for the two groups (the values of $\tilde{F}$ are obtained using the same cross-validation procedure). You may note that it is higher for the ASD group than for TD group. We can interpret these results as the fact that ASD samples tend to be more similar to the average than TD samples.

6.4. Other Distance-Based Classifiers

Figure 4 shows the performance of the logistic regression classifier with l2 regularization on the eigenvalues with a sliding window. Regularization parameter is set to 1 (other values showed very similar plots). The performance is not really stable throughout the position of the window. Nevertheless, the performance is slightly higher (mean ROC AUC 0.671) than the baseline.

Figure 5 shows the performance of the linear classifiers discussed in 5.4 that use the Euclidean, cosine and chi-squared distances. You can note that unlike the results of logistic regression, the performance of these classifiers changes smoothly depending on the position of the sliding window; there is a wide plateau on all three plots that corresponds to the sliding window set at the central part of the vector of sorted eigenvalues.

The best results obtained with these three classifiers are quite close: mean ROC AUC 0.676, 0.664 and 0.685 for Euclidean, cosine and chi-squared distances, respectively. A slightly better result is achieved with the chi-squared distance. Still, these results are worse

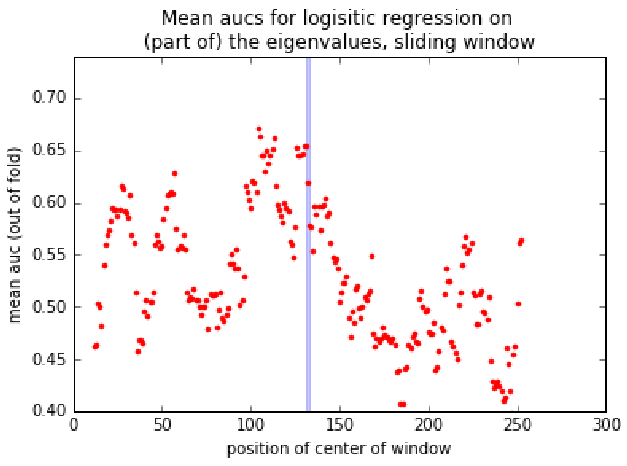

Figure 4. Logistic regression on Ω, mean ROC AUC values depending on the position of the sliding window

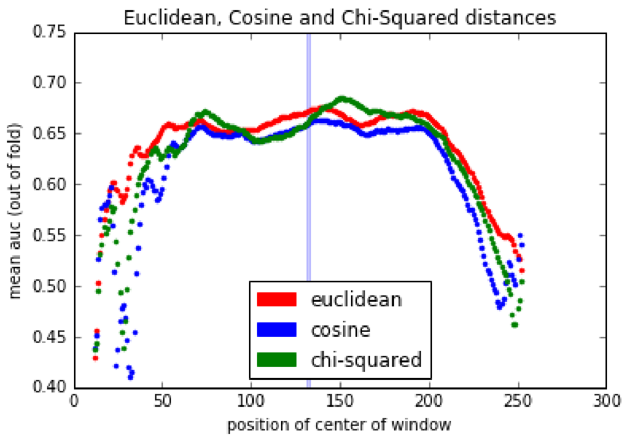

Figure 5. Linear classifiers that use Euclidean, cosine and chi-squared distances from the average, mean ROC AUC

than those obtained with our proposed method (although somewhat better than the baseline). Note that the standard deviations of ROC AUC values (shown in Table 1) are much higher than those obtained with the Dirichlet-based classifiers.

7. Conclusions

In this paper, we proposed a novel algorithm for classifying brain networks. We started with eigenvalues of the normalized graph Laplacians as initial feature vectors and constructed a linear classifier that used an assumption that these eigenvalues were sampled from a Dirichlet distribution. We demonstrated how the proposed algorithm worked in a classification task of separating autism spectrum disorder from typical development. The classifier showed good performance, indicating that graph spectra were informative for connectome classification tasks.

Similar to our previous works, we used the eigenvalues of the normalized graph Laplacians to construct a classifier. Unlike our previous works, the vector of eigenvalues was assumed to be sampled from the Dirichlet distribution. A linear classifier based on a single feature was proposed. The feature was constructed by estimating the parameter of the average distribution and then computing the likelihood of each sample given this parameter. Additionally, only part of the eigenvalues was taken into account by introducing a window and sliding it through all possible positions. The best classifier constructed with this procedure showed a mean ROC AUC 0.73. Importantly, the proposed classifier was robust to variation in the data and showed vary low standard deviation of ROC AUC value across different splits of the data into test and train parts within the cross-validation procedure.

For the purpose of comparison, we considered similar classifiers that used geometric distances instead of our proposed statistical method. These classifiers showed worse performance, implying that the Dirichlet distribution might be a useful model for the analysis of graph spectra in brain connectome classification tasks. It is also noteworthy that optimizing the position of the sliding window largely improved the performance of the original classifier. This fact points to the supposition that some part of the spectrum of the normalized graph Laplacian might be more informative than the rest of the spectrum for the classification task discussed in this paper.

Acknowledgments. The article was prepared within the framework of the Academic Fund Program at the National Research University Higher School of Economics (HSE) in 2016 (grant #16-05-0050) and by the Russian Academic Excellence Project "5-100". We gratefully acknowledge UCLA Multimodal Connectivity Database for making the dataset available to the research community.

References

[1] Dodonova, Y., Korolev S., Tkachev A., Petrov, D., Zhukov, L., Belyaev, M.: Classification of structural brain networks based on information divergence of graph spectra. Machine Learning for Signal Processing Proceedings (2016, accepted).

[2] Dodonova, Y., Belyaev, M., Tkachev, A., Petrov, D., Zhukov, L. : Kernel classification of connectomes based on earth mover's distance between graph spectra. Proceedings of the Workshop on Brain Analysis using Connectivity Networks (2016, accepted)

[3] Banerjee A., Jost J.: Graph Spectra as a Systematic Tool in Computational Biology.Discrete Applied Mathematics, 157, 2425–2431 (2009)

[4] Chung, F.: Spectral Graph Theory (1997)

[5] Mieghem, P.: Graph Spectra for Complex Networks (2011)

[6] Frigyik, B., Kapila, A., Gupta, M.: Introduction to the Dirichlet Distribution and Related Processes (2010)

[7] Minka, T.: Estimating a Dirichlet distribution (2012)

[8] Available online at: https://github.com/ericsuh/dirichlet

[9] Brown, J.A., Rudie, J.D., Bandrowski, A., Van Horn, J.D., Bookheimer, S.Y.: The UCLA multimodal connectivity database: a web-based platform for brain connectivity matrix sharing and analysis. Frontiers in Neuroinformatics 6, 28 (2012)

[10] Rudie, J.D., Brown, J.A., Beck-Pancer, D., Hernandez, L.M., Dennis, E.L., Thompson, P.M., et al.: Altered functional and structural brain network organization in autism. Neuroimage Clin 2, 79–94 (2013)

[11] Petrov, D., Dodonova, Y., Zhukov, L., Belyaev, M.: Boosting Connectome Classification via Combination of Geometric and Topological Normalizations. PRNI Proceedings (2016, accepted)