JMLR: Workshop and Conference Proceedings 46:67–76, 2015 ECML 2014

Neural Connectivity Reconstruction from Calcium Imaging Signal using Random Forest with Topological Features

Wojciech M. Czarnecki WOJCIECH.CZARNECKI@UJ.EDU.PL
Faculty of Mathematics and Computer Science
Jagiellonian University, Krakow, Poland

Rafal Jozefowicz RAFALJ@GOOGLE.COM
Google inc., New York, USA

Editors: Demian Battaglia, Isabelle Guyon, Vincent Lemaire, Jordi Soriano

Abstract

Connectomics is becoming an increasingly popular area of research. With the recent advances in optical imaging of the neural activity tens of thousands of neurons can be monitored simultaneously. In this paper we present a method of incorporating topological knowledge inside data representation for Random Forest classifier in order to reconstruct the neural connections from patterns of their activities. Proposed technique leads to the model competitive with state-of-the art methods like Deep Convolutional Neural Networks and Graph Decomposition techniques. This claim is supported by the results (5th place with 0.003 in terms of AUC ROC loss to the top contestant) obtained in the connectomics competition organized on the Kaggle platform.

Keywords: Random forest, neural connectivity reconstruction, connectome, topological features

1. Introduction

The study of connectomes is becoming an increasingly popular area of research. With the recent advances in optical imaging of the neural activity tens of thousands of neurons can be monitored simultaneously. We are trying to solve an inverse problem: reconstruct the original direct connections between neurons based on their patterns of activities.

Many tools have been proposed for the neural connectivity reconstruction based on their time series activity. Used approaches range from simple coefficients computed for each pair of signals, like Transfer Entropy (TE) (Honey et al., 2007) and its further modification – Generalized Transfer Entropy (GTE) (Stetter et al., 2012) to machine learning approaches designed specifically for this kind of problem, like Friedman et al. Graphical Lasso model (Friedman et al., 2008). The proposed method exploits multiple causality indices to reconstruct the graph with use of the topological features to indirectly achieve the similar task to Graphical Lasso. Instead of modifying or developing a new method, we show how one can manipulate the data representation to make it possible to achieve good results with use of a standard classification method – Random Forest (Breiman, 2001).

The rest of the paper is structured as follows. Section 2 describes in detail the methods used by the proposed approach. In Section 3 we show the methodology used for testing

the experiments and the comparison of the results. Finally, in Section 4 we summarize our findings and the properties of the proposed solution.

2. Methods

In the proposed approach, we deal with the neural connectivity reconstruction from the neurons' activity time series by modeling it as a simple binary classification problem. Each pair of neurons is represented by a constant size real vector, where each dimension is one feature value. In other words, given time series of i'th neuron x_i we create a point

$$p_{ij} = [f_1(x_i, x_j), f_2(x_i, x_j), \cdots, f_n(x_i, x_j)]^T \in \mathbb{R}^n,$$

such that each f_k is one feature extractor. For simplicity we use the notation $f_k(i, j) := f_k(x_i, x_j)$. Our training set consists of pairs (p_{ij}, c_{ij}) where c_{ij} equals 1 if there is a directed edge between i'th and j'th neuron, and 0 otherwise.

We have chosen Random Forest (RF) (Breiman, 2001) as the underlying classification tool due to its high efficiency and parallelism. We are dealing with the classification problem, where training set consists of millions of points in the input space with more than a hundred dimensions. Training other non-linear models (like kernelized SVMs, or even KNN) take significantly more time. Unfortunately, there are some limitations, which make application of this model difficult for a considered problem. Most importantly, RF in its basic, most efficient implementation, is unable to use any combination of features to make a decision. As a result, it is sensitive to relations as simple as linear combinations of the dimensions. For example, adding $f_i - f_j$ can significantly improve the classification process. This also explains our focus on adding various feature transformations to the input space that would help the classifier make better decisions. We evaluated many of them and will go into more detail about the process in the later subsections.

There are two basic types of features extractors used:

- we use existing, well known features including correlation coefficient or generalized transfer entropy,
- we introduce topological features, which encode the various graph's structure information in our representation.

Now we briefly describe both types of features.

2.1. Efficient features extraction

The dataset for a single network consists of a time series of neural activities for each of the 1000 neurons. The data itself is simulated but should closely resemble real recordings of cultured neurons. This allows us to have ground truth of connections, which is desirable for evaluation purposes. We are given a one hour of recording with 50Hz frequency (approximately $180,000$ data points per neuron) and the goal of the problem is to determine, for each pair of neurons, whether there is a direct connection between them.

It is worth noting that, in our problem, the average length of neuron's spike is shorter than the available frame rate of captured activities. This is a property of the simulator to model the limited time resolution and not allowing to easily separate individual spikes.

Because of that, we can safely assume that if the inferred spikes occur more than 2 frames away, they can be considered independent (not directly influenced).

A simple pre-processing scheme was applied to the raw data in order to retrieve spike times of the neurons. We looked at the series of the differences of the two consecutive elements and put them into 2 or 3 buckets (which gave us one level of parameterization) based on the chosen quantiles from overall network data. Furthermore, as suggested in (Stetter et al., 2012), we filtered out the points in time in which the mean neural activity exceeded the chosen threshold.

We have used several different base predictors that were computed on each pair of the neural activities (pre-processed):

- Cross-correlation (XC), $E[(X - \overline{X})(Y - \overline{Y})]$
- Cross-correlation with a lag of 1 frame (XC-L1)
- Generalized Transfer Entropy (GTE), $H(Y^t|Y^{t-1}, Y^{t-2}) - H(Y^t|Y^{t-2}, Y^{t-1}, X^{t-1}, X^t)$, where $H(X)$ is a Shannon entropy of X
- Information Gain, $G(Y^t|Y^{t-1}) - G(Y^t|X^t, Y^{t-1})$, where $G(X)$ is a Gini index (IGG) or a Shannon entropy of X (IGE)

One notable property of the above metrics is that we compare the values of the two series at the same point time (Instant Feedback Term - IFT) with the exception of XC-L1. This is important with the limited resolution of the raw data as the neurons often appear to spike at the same time on the recording. IFT allows us to capture this information.

With the 5 base algorithms, we considered 2 ways of splitting the raw data into buckets and 12 different mean activity thresholds. We used all 120 possible combinations as features for RF and let the learning algorithm find the best parameters.

Since this is a computationally intensive task we have developed efficient implementations of the algorithms in Python with the critical paths written in Cython. A single method requires 600MB of RAM and about 20 minutes of a single CPU time using modern hardware, which is orders of magnitude faster than the computation times mentioned in (Stetter et al., 2012). Furthermore, because of its low memory footprint, the approach easily scales linearly with the number of available CPU cores. This allowed us to iterate quickly and evaluate many potential features.

It is also worth mentioning that we have tried many other candidates as base predictors, including lagged and weighted correlations, Granger causality and GTE on the reversed time series. We did not notice any significant improvements when using them along with the chosen features and realized that adding more feature transformations gives us much higher impact.

Table 1 summarizes the performance of the base predictors. The information gain metric with Gini index turns out to be used the most in the final model. One explanation of the poor performance of XC-L1 is the fact that it's missing the information about the spikes that appear to occur at the same time (by the definition of the metric). As for GTE, it might be the case that it's looking at the values too far out in the past (depends on the values of three consecutive frames at any given point), which is adding more noise to the predictor. It could have been more successful if the frame rate was higher. The other explanation could be that, as noted in (Stetter et al., 2012), with the optimal conditioning level IGE/IGG can

base feature	importance	TOP10	TOP50	TOP100
IGG	0.62	9	26	51
IGE	0.18	1	9	27
XC	0.15	0	15	20
XC-L1	0.04	0	0	1
GTE	0.01	0	0	1

Table 1: Analysis of the base features in the final model. TOPX denotes how many features in top X most important features were using the particular base predictor.

out-perform GTE. Random Forest might be doing a good job in deciding which threshold is the best for a given network resulting in the superior results of the information gain methods.

2.2. Random Forest

Random Forest (Breiman, 2001) (RF) is a very successful ensemble learning technique for both classification and regression problems using decision (or regression) trees. The main idea behind this algorithm is to perform a bagging (bootstrap aggregation) together with a random subspaces method. Given a constant B (size of the forest), the tree bagging performs the following steps:

1. for $k = 1$ to B
   (a) sample with replacement N training samples from $T = \{(p_{ij}, c_{ij})\}_{i,j}^{N}$ to create T_i.
   (b) train a decision tree on T_i and call the resulting predictor $t_k : \mathbb{R}^n \to \{0, 1\}$.
2. return predictor

$$t(p) = \arg \max_{c \in \{0,1\}} v(c, p),$$

where $v(c, p) = |\{t_k(p) = c : k \in \{1, \ldots, B\}\}|$.

The only modification that is used in Random Forest is to train a non-classical decision tree in step 1. (b), namely, at each node split the random subset of features is selected to be considered. This small modification adds an important regularization to the whole training process, so the valid choice of the size of this random subset is crucial. For classification procedure a well known heuristic is to chose a number of features equal to the square root of the input space. In our solution we tuned this parameter in order to find the most appropriate size of the random subspaces and found out that in this particular problem the optimal value is a bit bigger (40).

The choice of such classifier was mainly motivated by:

- its highly parallel structure – each t_k can be independently trained and evaluated,
- fast training time as the decision trees are very simple classifiers,

- existence of efficient, free implementations like the one in scikit-learn Pedregosa et al. (2011),
- giving a direct estimation of the classification confidence $\hat{P}(c|p) = v(c,p)/B$.
- dealing well with missing features (not the case in our scenario, but could be helpful in general as some statistics may be undefined in border cases).

Unfortunately, RFs have some drawbacks, namely:

- they consist of many metaparameters that have to be tuned,
- they are unable to express any, even linear, combinations of features in general (some modifications, like Rotation Forests (Rodriguez et al., 2006) have been proposed but they are much less efficient).

Both of these problems are solved (to some extent) by introduction of the specific topological features and a heuristic method of training described in the further sections.

2.3. Random Forest with Topological Features

The true input data has a graphical form, which means that there are important relations encoded in the mutual location of particular neurons. For example, for the exact causality detection it is not only important how high are some causality-related features between i'th and j'th neuron, but also what are the same features computed in the opposite direction. This leads to the need of incorporating features into the data representation, which include graph structure and can be directly used by the RF. During heavy testing we have developed the set of topological features, which significantly increase the model's quality.

For each particular feature extractor f (being for example GTE with threshold 0.12) we considered:

- Normalized difference $\frac{f(i,j)-f(j,i)}{f(i,j)+f(j,i)}$, in order to detect the edge's direction it is important to know whether the opposite direction is more or less likely. The normalization helps to work with wide range of possible $f(i,j)$ values in a uniform way.
- Geometrical closure $\max_k \sqrt{f(i,k)f(k,j)}$, in order to check whether the high "causality" between i'th and j'th neuron is a result of an existence of k'th neuron, through which the signal actually flows.
- Markov closure $\sum_k f(i,k)f(k,j)$, assuming that we view the whole problem as a kind of Markov process, where states are describing which neuron spikes, we can compute the overall probability of going from i'th neuron spike to j'th neuron spike, if $f(i,j)$ is a transition probability.
- Feature ratios $\frac{f(i,j)}{\max_k f(k,j)}$, $\frac{f(i,j)}{\max_k f(i,k)}$, $\frac{f(i,j)}{\max_k f(i,k)f(k,j)}$, $\frac{f(i,j)}{\sum_k f(i,k)f(k,j)}$ which were included due to the limitations of the RF model that was unable to divide two existing features by themselves.
- Scaled ratios $\frac{f(i,j)}{\sqrt{\sum_k f(i,k)f(k,j)}}$, $\frac{f(i,j)}{(\sum_k (f(i,k)f(k,j))^{3/2})^{1/3}}$, $\frac{f(i,j)}{\max_k \sqrt{f(i,k)f(k,j)}}$ to neglect some scale issues.

- Network Deconvolution: $F \cdot (I + F)^{-1}$ inspired by Feizi et al. (2013), where F is a matrix such that $F_{ij} = f(i, j)$. If we treat F as a transitive closure of some matrix G such that $\lim_{n\to\infty} G^n = 0$ (with some additional assumptions), then the given formula is an inverse operation to $F = G + G^2 + \cdots = G \cdot (I - G)^{-1}$. The intuition behind is that, in some sense, G captures the direct relationships between the neurons.

It is worth noting, that each such topological feature adds as many new dimensions, as there are non-topological features[1] in the representation. So, for example, if we have 4 types of features extractors, each with 6 types of some parameters (thresholds), then each topological feature actually adds $4 \times 6 = 24$ new dimensions. However, adding all of such features would significantly increase the input space size and lead to the weaker classifier due to the *curse of dimensionality*. Luckily, RF can be used in a greedy, closed loop manner, which enables us to iteratively change the data representation.

2.4. Random Forest training with constant representation changes

In order to deal with the increasing number of dimensions as well as the fact, that model's metaparameters (number of minimum samples in a leaf, maximum number of features considered, etc.) heavily depend on the input space dimensionality we propose the following closed loop process for each new topological features f' to be considered and for a representation ϕ with a model currently scoring s_ϕ under some metric (like in our case AUC ROC score).

1. take next f',
2. add f' to current representation ϕ forming $\phi^+_{f'}$,
3. train forest on the whole dataset using $\phi^+_{f'}$.,
4. Compute the *relative feature importances* (averaged expected fraction of the samples features contribute to) for each $f \in \phi^+_{f'}$ and drop as many worst dimensions, as there were added by f' forming $\phi_{f'}$,
5. evaluate forest using leave-one-out cross validation using $\phi_{f'}$ getting score $s_{\phi_{f'}}$,
6. if $s_{\phi_{f'}} > s_\phi$ then $\phi \leftarrow \phi_{f'}, s_\phi \leftarrow s_{\phi_{f'}}$,
7. goto 1.

It is easy to see that such procedure ensures constant size representation, which reduces the problem of the high dimensionality and the requirement of constant metaparameters refitting.

Table 2 summarizes the results of the proposed methodology by providing final importance of each topological feature in our Random Forest. One interesting phenomenon is a very high usage of the Minkovsky-like scaling of the Markov closure based ratio. This may be a result of the minor scaling issues (each network has a bit different values, this Minkovsky scaling makes the differences much less significant).

1. Except normalized difference which is also applied for all non-symmetrical topological and non-topological features.

topological feature	equation	importance	TOP10	TOP50	TOP100
no topology	$f(i,j)$	0.01	0	0	0
normalized difference	$\frac{f(i,j)-f(j,i)}{f(i,j)+f(j,i)}$	0.06	0	0	4
geometrical closure	$\max_k \sqrt{f(i,k)f(k,j)}$	0.09	0	4	16
markov closure	$\sum_k f(i,k)f(k,j)$	0.05	0	1	18
feature ratios 1	$\frac{f(i,j)}{\max_k f(k,j)}$	0.01	0	0	1
feature ratios 2	$\frac{f(i,j)}{\max_k f(i,k)}$	0.15	1	5	14
feature ratios 3	$\frac{f(i,j)}{\max_k f(i,k)f(k,j)}$	0.03	0	0	6
feature ratios 4	$\frac{f(i,j)}{\sum_k f(i,k)f(k,j)}$	0.04	0	1	18
scaled ratios 1	$\frac{f(i,j)}{\sqrt{\sum_k f(i,k)f(k,j)}}$	0.23	3	14	18
scaled ratios 2	$\frac{f(i,j)}{(\sum_k (f(i,k)f(k,j))^{3/2})^{1/3}}$	0.46	6	26	33
scaled ratios 3	$\frac{f(i,j)}{\max_k \sqrt{f(i,k)f(k,j)}}$	0.05	0	4	10

Table 2: Analysis of the features in the final model. TOPX denotes how many features in top X most important features were using the particular topology out of the total 535 best features chosen in the procedure described earlier.

3. Evaluation

The proposed method was evaluated during the CONNECTOMICS[2] competition organized by Challenges in Machine Learning[3] and hosted on the Kaggle[4] platform. The duration of the contest was 3 months in which the participants were required to submit solutions in the csv format for the 2 networks. The submitted file should consist of one real number for each pair (ordered) of neurons of the two held out networks, representing the confidence of whether there is a directed connection between the given nodes. After each submission, the participants would receive an instant feedback with the score on one of the held out networks (Valid). To prevent the overfitting the solutions were ranked based on the scores of the second network (Test) and the final results were only available after the competition ended. The metric used for models' evaluation was an area under the ROC curve, which is a standard metric in such tasks (Stetter et al., 2012).

We used code written mainly in Python (with Cython elements) with the use of the NUMPY, SCIPY and SCIKIT-LEARN (Pedregosa et al., 2011) libraries. The tests were performed on the Fermi supercomputer consisting of 64 computational units (eight 8-core processors) to exploit the parallel nature of used model. However, it would be also possible to run the whole solution directly on the GPU units as feature extractors, topological features and RF training itself can be easily done on the GPU unit (Van Essen et al., 2012).

The dataset used for the evaluation purposes consists of a time series of four $(1,000$ neurons each) networks activity. Additional two networks were held out by organizers,

2. http://www.kaggle.com/c/connectomics/
3. http://www.chalearn.org/
4. http://www.kaggle.com

so they were not used during internal evaluation, however, the scores of our final model are available. Due to our approach, this gives $4,000,000$ samples of a binary classification problem.

All metaparameters were fitted using leave-one-out cross validation. We considered each network as one "set of samples", and performed LOO on this level of granularity (so there were 4 iterations, in each we trained on $3,000,000$ points and tested on $1,000,000$, all coming from never seen before network). The process of adding topological features was performed according to the scheme from the previous section.

model (**team name at Kaggle**)	features	LOO	Valid	Test
Logistic Regression	basic	0.90245	-	-
SVM	basic	0.90305	-	-
Random Forest	basic	0.90910	-	-
Random Forest	+normalized difference	0.91326	0.91977	0.91720
Random Forest	+geometrical closure	0.92635	0.92795	0.92757
Random Forest	+markov closure	0.93062	0.93239	0.93250
Random Forest	+feature ratios	0.93597	0.93618	0.93634
Random Forest (**Lejlot & Rafal**)	+scaled ratios	0.93781	0.93761	0.93826
Random Forest	+network deconvolution	0.94224	0.94239	**0.94269**
Deep CNN (**Lukasz 8000**)	raw signal	-	0.93920	0.93956
Partial Correlations (**AAAGV**)	-	-	0.94262	0.94161

Table 3: Results of different models under AUC ROC metric for leave-one-out cross validation on 4 networks (LOO), hold-out valid (Valid) and test (Test) networks.

Table 3 summarizes the results obtained for the considered models (team Lejlot & Rafal) as well as two competitors' approaches. One of them is a Deep Convolutional Neural Network (Sainath et al., 2013) trained on the pure signal (team Lukasz 8000) and the second one is a Partial Correlation Estimation Model based on (De La Fuente et al., 2004) (team AAAGV).

It is worth noting that our simple, RF based approach, where the model is not well suited neither for the time series processing nor for the graph reconstruction, behaves very well. In particular, it achieves about $0.001 - 0.002$ worse results than a deep neural network, being at the same time much simpler model and implemented in dozens of existing libraries. On the other hand it achieves score around 0.003 (result on the Valid set seems to be overfitting due to the great decrease in the result on the test network) from the partial correlations model.

Our cross-validation scheme estimates the score on hold-out networks really well, suggesting that this RF solution is not overfitting the training data. It was interesting to see that while we were adding more features, the LOO estimations were getting closer to the scores on the unseen networks (as can be observed from the table).

It seems to be another proof of wide applicability of Random Forest model, which after adding topological features is able to compete head-to-head with state-of-the-art methods.

The proposed method achieved 5th place in the competition (losing about 0.003 in terms of AUC ROC). However, among all best scoring teams, only the proposed approach was

using different model from Partial Correlations, Network Deconvolution and Deep Convolutional Neural Network.

After the competition ended, we analyzed the performance of a simplified version of the Network Deconvolution algorithm for a feature extractor, as suggested by one of the contestants. It improved our solution further and beating the top result from the competition by approximately 0.001.

4. Conclusions

To the best of our knowledge using a binary classifier with topological features is a different approach than taken by other competitors and to what we could find in the existing literature. It has also some interesting properties:

- The feature generation and training / prediction of the Random Forest algorithm is fully parallelizable and it scales linearly with the number available of processors (up to the point of training one tree, which is on the order of minutes).
- The prediction for the edge between nodes i and j depends only on the metrics for neighbors of i and j. Due to its local nature it has potential to be applicable for much larger graphs since we are not required to keep the whole graph in memory. We just need to be able to compute metrics for the neurons that are close enough to i and j.
- It makes very little assumptions about the underlying neurobiological setting of the problem and it is based only on the simple characteristics of the data (like correlations of neural activities). In fact, the only part that is related to neuron activities is the simple initial filtering and pre-processing. The model behaves very well even though the spike inference is very basic and far from perfect. It is also easy to explain and implement.
- The method does not assume the graph to be undirected as opposed to graphical lasso approach and thus can extract directed relationships between neurons.

References

Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

Alberto De La Fuente, Nan Bing, Ina Hoeschele, and Pedro Mendes. Discovery of meaningful associations in genomic data using partial correlation coefficients. *Bioinformatics*, 20(18): 3565–3574, 2004.

Soheil Feizi, Daniel Marbach, Muriel Médard, and Manolis Kellis. Network deconvolution as a general method to distinguish direct dependencies in networks. *Nature biotechnology*, 2013.

Jerome Friedman, Trevor Hastie, and Robert Tibshirani. Sparse inverse covariance estimation with the graphical lasso. *Biostatistics*, 9(3):432–441, 2008.

Christopher J Honey, Rolf Kötter, Michael Breakspear, and Olaf Sporns. Network structure of cerebral cortex shapes functional connectivity on multiple time scales. *Proceedings of the National Academy of Sciences*, 104(24):10240–10245, 2007.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *The Journal of Machine Learning Research*, 12:2825–2830, 2011.

Juan José Rodriguez, Ludmila I Kuncheva, and Carlos J Alonso. Rotation forest: A new classifier ensemble method. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 28(10):1619–1630, 2006.

Tara N Sainath, Abdel-rahman Mohamed, Brian Kingsbury, and Bhuvana Ramabhadran. Deep convolutional neural networks for lvcsr. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, pages 8614–8618. IEEE, 2013.

Olav Stetter, Demian Battaglia, Jordi Soriano, and Theo Geisel. Model-free reconstruction of excitatory neuronal connectivity from calcium imaging signals. *PLoS computational biology*, 8(8):e1002653, 2012.

Brian Van Essen, Chris Macaraeg, Maya Gokhale, and Ryan Prenger. Accelerating a random forest classifier: Multi-core, gp-gpu, or fpga? In *Field-Programmable Custom Computing Machines (FCCM), 2012 IEEE 20th Annual International Symposium on*, pages 232–239. IEEE, 2012.