JMLR: Workshop and Conference Proceedings 46:37–44, 2015 ECML 2014

# Supervised Neural Network Structure Recovery

**Ildefons Magrans de Abril** Ildefons.Magrans.de.Abril@vub.ac.be
**Ann Nowé** Ann.Nowe@vub.ac.be
*Artificial Intelligence Lab, Vrije Universiteit Brussel, Pleinlaan 2, 1050 Brussels, Belgium*

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

## Abstract

This paper presents our solution to the European Conference of Machine Learning Neural Connectomics Discovery Challenge. The challenge goal was to improve the performance of existing methods for recovering the neural network structure given the time series of neural activities. We propose to approximate a function able to combine several connectivity indicators between neuron pairs where each indicator is the result of running a feature engineering pipeline optimized for a particular noise level and firing synchronization rate among neurons. We proved the suitability of our solution by improving the state of the art prediction performance more than 6% and by obtaining the third best score on the test dataset out of 144 teams.

**Keywords:** Neural Network, Structure Recovery, Causality, Time Series

## 1. Introduction

Most advanced neuroimaging techniques, based on calcium sensitive organic dyes (Tsien, 1981), are able to capture the *in vivo* activity of thousands of neurons. This represents a huge improvement with respect to the highly invasive neurophysiology multi-electrode recording tools barely capable of recording on the order of 100 neurons. The full potential development of this new instrumentation still requires analysis tools able to infer the underlying topology based on time-series of neuronal activity. The goal of the European Conference of Machine Learning Neural Connectomics Challenge (ChaLearn, 2014) was to encourage the application of machine learning techniques to recover the neural network structure using neural activity time-series recordings as input. This paper presents our method[1] to train a regression model that can predict the connectivity between neuron pairs given the time series of neural activities obtained from fluorescence signals. With this solution we obtained the third best score on the test dataset out of 144 teams.

The challenge setup consists of several neural network datasets for both training and testing purposes, an evaluation process, detailed information about the problem and sample code to get started. Neural network datasets consists of one hour time series of neural activities obtained from fluorescence signals sampled at $20ms$ intervals with values normalized in the interval $[0, 1]$, information about the position of each neuron in a square area of $1mm^2$ and the inter-neuron connectivity labels. The setup evaluation process is built upon the Area Under the ROC Curve (AUC) as evaluation metric and a real-time leaderboard showing the ranking of all teams according to the score obtained on an evaluation dataset.

1. https://github.com/ildefons/connectomics.

The score on a separated test dataset defined the final ranking of teams. This score was not visible and therefore could not be used to overfit the final model.

There are three major difficulties to detect connectivity among neuron pairs (Stetter et al., 2012): 1) episodes of synchronous bursting conveying low connectivity information, 2) a typical frame video rate is 20ms which is slower than the neuron's firing dynamic by one order of magnitude and 3) the background has noise.

The challenge sample code has two non-supervised methods: a simple solution based on correlation and a state of the art solution based on the Generalized Transfer Entropy (GTE) indicator (Orlandi et al., 2014; Stetter et al., 2012). The correlation based solution computes the connectivity indicators among 1000 neurons in few minutes without taking into account the causal direction. This naive example was meant uniquely to get as many people as possible started. The C++ implementation of the GTE-based solution is a directional connectivity indicator and it runs in about 12 hours on a high-end server. This second solution requires a careful parameter selection and it is a good example of how to score directed connections between neuron pairs taking into account the three major difficulties.

## 2. Model

The main observation that guided our work was that optimizing a single connectivity indicator, as suggested by the state of the art, may be a limiting strategy because it will tend to work optimally just on a particular regime (i.e. noise level and firing synchronization rate among neurons). Therefore, a function approximation able to optimally combine several indicators, computed using different parameters, should deliver an enhanced performance. This section describes the details of our solution. It consists of a feature engineering pipeline to compute the many connectivity scores according to different parameter values and a suitable model fitting strategy able to combine these features. Figure 1 shows the main components of our solution. The following sub-sections describe in detail each of the building blocks.

### 2.1. Spike inference

The first step in our feature engineering pipeline is the spike inference module. It is responsible for inferring spike trains out of time series of neural activities. We have evaluated two approaches: a naive method based on computing the difference between any two consecutive time steps, and a state of the art method (Vogelstein et al., 2009) based on the sequential Monte Carlo framework (a generatlization of the Baum-Welch algorithm to fit Hidden Markov Models) that finds the probability of the neuron spiking in each time step.

In both cases, spike trains are post-processed to remove background noise. A parameter named Noise Level (NL) defines the lower limit of a valid spike. Spikes below NL are zeroed. The evaluation of both spike inference methods was performed using the complete solution pipeline outlined in Figure 1 on a training network ($\text{Normal}_1$) constructed similarly to the test network. We analyzed the performance delivered by each method using many different parameters. We finally chose the Monte Carlo framework based method because it delivered a maximum AUC of 0.932 and an average AUC over all parameters of 0.909 while the naive method delivered a maximum AUC of 0.929 and an average AUC over all parameters of 0.902.

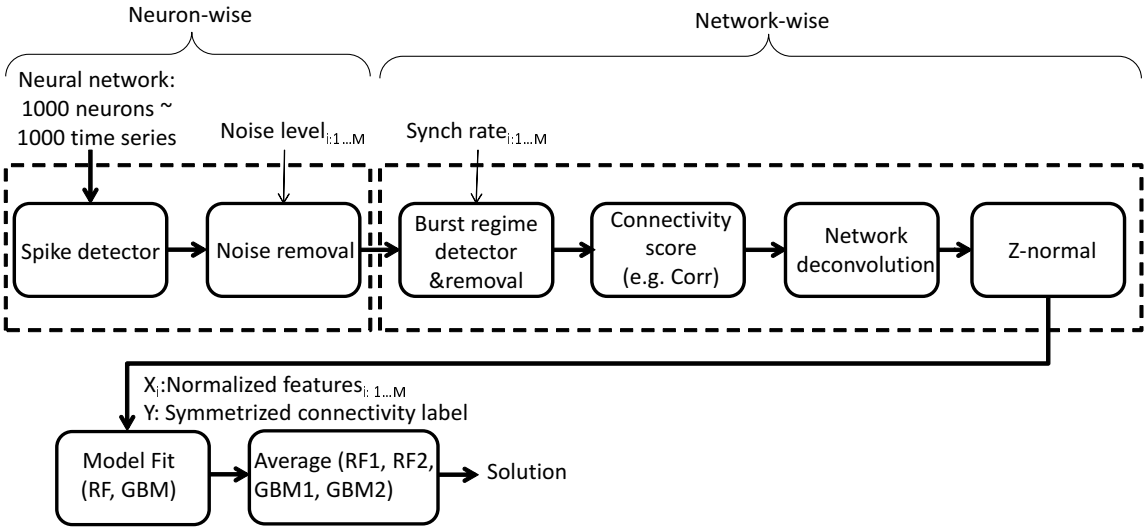


Figure 1: Solution block Diagram. It consists of a feature engineering pipeline to compute several connectivity indicators between each neuron pairs and a model fitting strategy to combine these features.

### 2.2. Connectivity indicators

Right after the noise removal module and before the computation of the inter-neuron connectivity indicators, the burst regime detector and removal module is responsible for identifying and removing time steps that contain a portion of neurons larger than the parameter Synchronization Rate (SR) firing at the same time. This step is required because only signals recorded during inter-burst periods convey elevated information about the neural net topology (Stetter et al., 2012). Formally, we remove all time steps $t$ such that $\sum_{i=1}^{N} \mathbb{1}\{ST_{it} \neq 0\} > SR.N$, where N is the number of neurons and $ST_{it}$ is the value of the spike train of neuron $i$ at time $t$ generated by the spike inference module described in Section 2.1.

We need to compute a connectivity indicator on each neuron pair and for each parameter combination (i.e. NL, SR), and all these computations have to be executed for several training networks, the evaluation and the test networks. Therefore, the computation efficiency is a key requirement for this module. Plain correlation was a suitable candidate according to this requirement. Performance-wise, correlation between spike trains delivered a performance equivalent to GTE between pairs of raw time series of neural activity (Orlandi et al., 2014; Stetter et al., 2012).

Correlation is a simple enough connectivity indicator to be able to compute the many indicators using reasonable resources. However, a limitation of using correlation as connectivity indicator is that it is unable to identify directed connections. Therefore, it may not be appropriate when the goal is to identify causal relationships. Fortunately, when the performance evaluation metric is the Area Under the ROC Curve, the added value of distinguishing the connectivity direction is low.

### 2.3. Network deconvolution

The final step of the feature engineering pipeline is the network deconvolution module. This module implements a recent network deconvolution algorithm (Feizi et al., 2013) meant to eliminate the combined effect of indirect paths of arbitrary length from an observed correlation matrix containing both direct and indirect effects. This step improves the quality of the connectivity indicators between neuron pairs by taking into account the whole connectivity matrix. This method has been able to improve the performance of state of the art solutions in other network reconstruction application scenarios (Feizi et al., 2013).

This method consists of the following steps: 1) to normalize in the interval [-1,1] the connectivity indication matrix described in Section 2.2, 2) to decompose with SVD the normalized matrix, 3) to compute the eigenvalues of the deconvolved matrix according to $\lambda_i^d = \frac{\lambda_i}{\lambda_i+1}$ with $\lambda_i$ being the $i_{th}$ eigenvalue of the normalized matrix, 4) to compose the direct dependency matrix according to $C_{dir} = UDU^{-1}$ where $U$ is a matrix of eigen-vectors and D is a diagonal matrix whose $i_{th}$ diagonal is $\lambda_i^d$.

A Z-normalization post-processing delivers a connectivity matrix with a distribution almost identical across different networks using the same parameters (i.e. NL, SR). This additional step is not part of the original deconvolution algorithm. It is motivated by the observation that the deconvolved matrix from any two different neural networks, computed with the same parameter set, had different distributions and therefore they cannot be used directly to train a supervised model.

The evaluation of the network deconvolution step was performed using again the complete solution pipeline outlined in Figure 1 on a training network ($\text{Normal}_1$) constructed similarly to the test network. We analyzed the performance using many different parameters. The deconvolution step improved the maximum performance from 0.911 to 0.932 and the average performance improved from 0.893 to 0.909.

### 2.4. Modeling approach

To overcome the limitation of using a single connectivity indicator optimized for a particular noise level and bursting synchronization rate, we propose to approximate a function able to combine several connectivity indicators between neuron pairs. More precisely, given a training network, the set of samples is defined by all possible combinations of different non-directed neuron pairs (e.g. N:Number of network neurons = 1000, number of samples $= \frac{N(N-1)}{2} = 499500$). Each sample consists of a set of connectivity indicators computed with the feature engineering pipeline described in the previous sections using different parameter values (i.e. NL, SR). Our modeling approach does not try to learn self-loops.

Fitting this function requires a method able to capture complex relationships among several very similar features. Gradient Boosting Machines (GBM) (Ridgeway, 2013) and Random Forest (Breiman et al., 2013) have been successfully used in other challenges with similar feature space complexity (Magrans de Abril and Sugiyama, 2013). Further concerns were the heterogeneity of the network topologies and the highly imbalanced training network datasets where only approximately 1% of neuron pairs were connected. We minimized the training network topology bias by 1) using a large minimum size for the tree leafs, and 2) averaging four models: two random forest fitted according to two training networks constructed similarly to the test network and two gradient boosting machines models fitted

according to the same two networks. We addressed the data imbalance problem by subsampling the training samples of non-connected neuron pairs down to 5 times the number of connected neuron pair samples.

## 3. Evaluation

During the model validation phase, we used three training networks ($\text{Normal}_1$, $\text{Normal}_2$ and $\text{Normal}_3$), the evaluation network and the test network. All networks consists of 1000 neurons and approximately 1% of neuron pairs were connected. According to the challenge data description, all these networks were constructed similarly. For each network we extracted a number of connectivity indicator matrices. Each connectivity matrix was computed running the feature engineering pipeline described in Section 2 with a given parameter set (i.e. NL,SR). More precisely, for each network we computed 252 matrices according to all possible parameters set combinations where:

$NL \in \{.07, .075, .08, .085, .09, .1, .11, .12, .13, .14, .15, .16, .17, .18\}$

$SR \in \{25, 50, 75, 100, 150, 200, 250, 300, 350, 400, 450, 500, 550, 600, 650, 700, 750, 800\}$

For each training network (e.g. $\text{Normal}_1$), we trained a RF and a GBM. To cope with inconsistencies among networks, we used a large minimum size for the tree leafs (450 for RF, 400 for GBM).

Table 1: Performance results when using the training networks for both model fitting and testing purposes. The row labels define the network under test and column labels define the predictive model (e.g. GTE $\sim$ Generalized Transfer Entropy, $\text{R}_1$+$\text{G}_1$ $\sim$ average of RF and GBM both fitted with $\text{network}_1$, ALL $\sim$ average of RF and GBM both fitted with the two other networks not being tested as proposed in Section 2.4).

| **T** | **GTE** | $\mathbf{R}_1$ | $\mathbf{G}_1$ | $\mathbf{R}_2$ | $\mathbf{G}_2$ | $\mathbf{R}_3$ | $\mathbf{G}_3$ | $\mathbf{R}_1$+$\mathbf{G}_1$ | $\mathbf{R}_2$+$\mathbf{G}_2$ | $\mathbf{R}_3$+$\mathbf{G}_3$ | **ALL** |
|---|---|---|---|---|---|---|---|---|---|---|---|
| $\mathbf{N}_1$ | 0.885 | NA | NA | .9392 | .9388 | .9394 | .9386 | NA | .9398 | .9398 | .9401 |
| $\mathbf{N}_2$ | 0.889 | .9399 | .9401 | NA | NA | .9403 | .9399 | .9409 | NA | .9409 | .9413 |
| $\mathbf{N}_3$ | 0.884 | .9393 | .9396 | .9392 | .9396 | NA | NA | .9402 | .9402 | NA | .9405 |

Table 1 shows the performance results when using the training networks for both model fitting and testing purposes. Column 1 shows the performance obtained with the state of the art solution (i.e. Generalized Transfer Entropy indicator (Orlandi et al., 2014; Stetter et al., 2012)). Column 2 to 7 show the prediction performance when we use a single RF or GBM trained with one training network. It shows that our supervised modeling approach is able to reliably deliver a superior performance compared to the best individual connectivity indicator. For instance, $\text{RF}_2$ to predict the connectivity of $\text{Normal}_1$ network delivers a prediction performance of .9392 while the best individual connectivity indicator computed with our feature engineering pipeline delivers a performance of 0.932 and an optimized connectivity indicator computed with GTE delivers a performance of 0.889.

Column 8 to 10 show the prediction performance when we average the predictions delivered by RF and GBM trained with one training network. They show experimental evidence that by averaging the predictions of several models we were able to further improve the prediction performance. From these 3 columns we can also conclude that there is not a

training network that provides a superior performance (e.g. training on $\text{Normal}_1$ and testing on $\text{Normal}_2$ delivers a performance of .9409 and training on $\text{Normal}_3$ and testing on $\text{Normal}_2$ delivers the same performance of 0.9409). Finally, column 11 shows the prediction performance when we average the predictions delivered by RF and GBM trained with the two other training networks as proposed in Section 2.4.

Table 2: Performance results when using the training networks for both model fitting and testing purposes. The row labels define the network under test and column labels define the the random seed used to sub-sample training samples of non-connected neuron pairs. All prediction models are computed according to Section 2.4.

| **Test** | **Seed**$_1$ | **Seed**$_2$ | **Seed**$_3$ |
|---|---|---|---|
| **Normal**$_1$ | .94007 | .94009 | .94009 |
| **Normal**$_2$ | .94134 | .94129 | .94121 |
| **Normal**$_3$ | .94051 | .94053 | .94048 |

Table 3 presents the performance results when using the training networks to fit a complete model according to the description of Section 2.4 (i.e. last column of Table 1) and using different random seeds before the sub-sampling of non-connected neuron pair samples. The row labels define the network under test and the column labels define the random seed applied before sub-sampling. It shows that sub-sampling has a very small effect on the model performance. Therefore, we are unlikely losing much information.

Finally, a model trained with $\text{Normal}_1$ and $\text{Normal}_2$ delivered a performance on the test network of .9406 which is .06 higher than the performance obtained with the best known solution before the challenge started (i.e. Generalized Transfer Entropy (GTE) indicator (Orlandi et al., 2014; Stetter et al., 2012)).

## 4. Conclusions and future work

Our modeling hypothesis is that by approximating a function able to optimally combine several indicators, computed using different parameters (i.e. noise level and firing synchronization rate among neurons), we could deliver an enhanced performance. The feature engineering pipeline is responsible for the computation of the connectivity indicators. It is based on a modular design able to separately address the different difficulties to detect connectivity among neuron pairs: 1) episodes of synchronous bursting conveying low connectivity information, 2) a typical frame video rate is $20ms$ which is slower than the neuron's firing dynamic by one order of magnitude and 3) the background has noise. We have proven the suitability of our solution by improving the state of the art prediction performance (AUC) in more than 6% and by obtaining the third best score on the test dataset out of 144 teams.

However, we believe that there is still room for improvement. For instance, important functional limitations of our model are that it is unable to identify the connectivity directions and self-loops. We also believe that there exist possibilities to improve our model such as using a finer grained grid of parameters or by using semi-supervised variants of RF (Leistner et al., 2009) and GBM (Dai et al., 2007).

Another limitation of our method is a high computational cost mainly due to the large number of connectivity indicators on several training networks and the test network. More precisely, the many correlation matrix and the SVD step during the network deconvolution have a computational complexity on the order of $O(MKNn^2)$ and $O(MKn^3)$ respectively, where n is the number of neurons (1000), N is the number of time steps (180000), K is the number of connectivity indicators (252) and M is the number of networks (3). Running on an i7 quad core laptop with 32 Gbytes of RAM, it takes 48 hours to compute the connectivity indicators for all networks, just below 5 hours to compute the spike trains and just above 2 hours to fit the random forest and the GBM models.

## Acknowledgments

Ildefons Magrans de Abril and Ann Nowé were supported by EU FP7 framework's Marie Curie Industry-Academia Partnerships and Pathways (IAPP) project SCANERGY, under grant agreement number 324321.

## Appendix A. Result table

Table 3: Summary table with team name, final private leaderboard performance and performance of the winner.

| | |
|---|---|
| **Team name** | Ildefons Magrans |
| **Private leaderboard performance** | 0.94063 |
| **Performance of the winner** | 0.94161 |

## References

Leo Breiman, Adele Cutler, Andy Liaw, and Matthew Wiener. R package randomforest: Breiman and cutlers random forests for classication and regression. Version 4.6-7, 2013.

ChaLearn. Connectomics challenge. http://connectomics.chalearn.org/, 2014.

Wenyuan Dai, Qiang Yang, Gui-Rong Xue, and Yong Yu. Boosting for transfer learning. In *Proceedings of the 24th international conference on Machine learning*, pages 193–200. ACM, 2007.

Soheil Feizi, Daniel Marbach, Muriel Mdard, and Manolis Kellis. Network deconvolution as a general method to distinguish direct dependencies in networks. *Nature Biotechnology*, 31:726–733, July 2013.

Christian Leistner, Amir Saffari, Jakob Santner, and Horst Bischof. Semi-supervised random forests. In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 506–513. IEEE, 2009.

Ildefons Magrans de Abril and Masashi Sugiyama. Winning the kaggle algorithmic trading challenge with the composition of many models and feature engineering. *IEICE Transactions on Information and Systems*, 96(3):742–745, 2013.

Javier G. Orlandi, Olav Stetter, Jordi Soriano, Theo Geisel, and Demian Battaglia. Transfer entropy reconstruction and labeling of neuronal connections from simulated calcium imaging. *arXiv:1309.4287v2*, May 2014.

Greg Ridgeway. R package gbm: Generalized boosted regression models. Version 2.1, 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), August 2012.

R.Y. Tsien. A non-disruptive technique for loading calcium buffers and indicators into cells. *Nature*, 290:527–528, April 1981.

Joshua T. Vogelstein, Brendon O. Watson, Adam M. Packer, Rafael Yuste, Bruno Jedynako, and Liam Paninski. Spike inference from calcium imaging using sequential monte carlo methods. *Biophysical journal*, 97(2):636–655, 2009.