ECML 2014, Neural Connectomics Workshop, 15 September 2014

Supervised Neural Network Structure Recovery

Ildefons Magrans de Abril

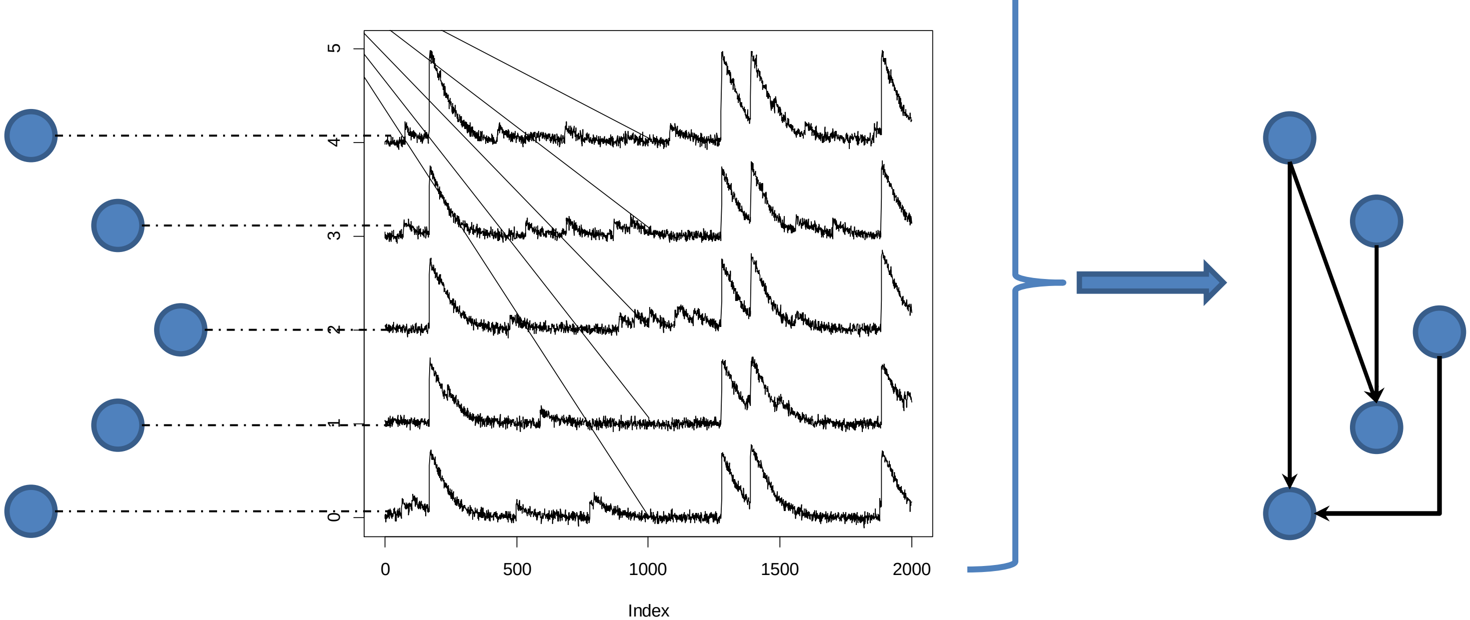
Vrije Universiteit Brussel

Challenge setup
Major difficulties
Solution guidelines
Design overview
Building blocks
Experiments
Conclusions

Challenge setup

Input: time series of neural activity obtained from fluorescence signals

Goal: to predict the connectivity between neuron pairs

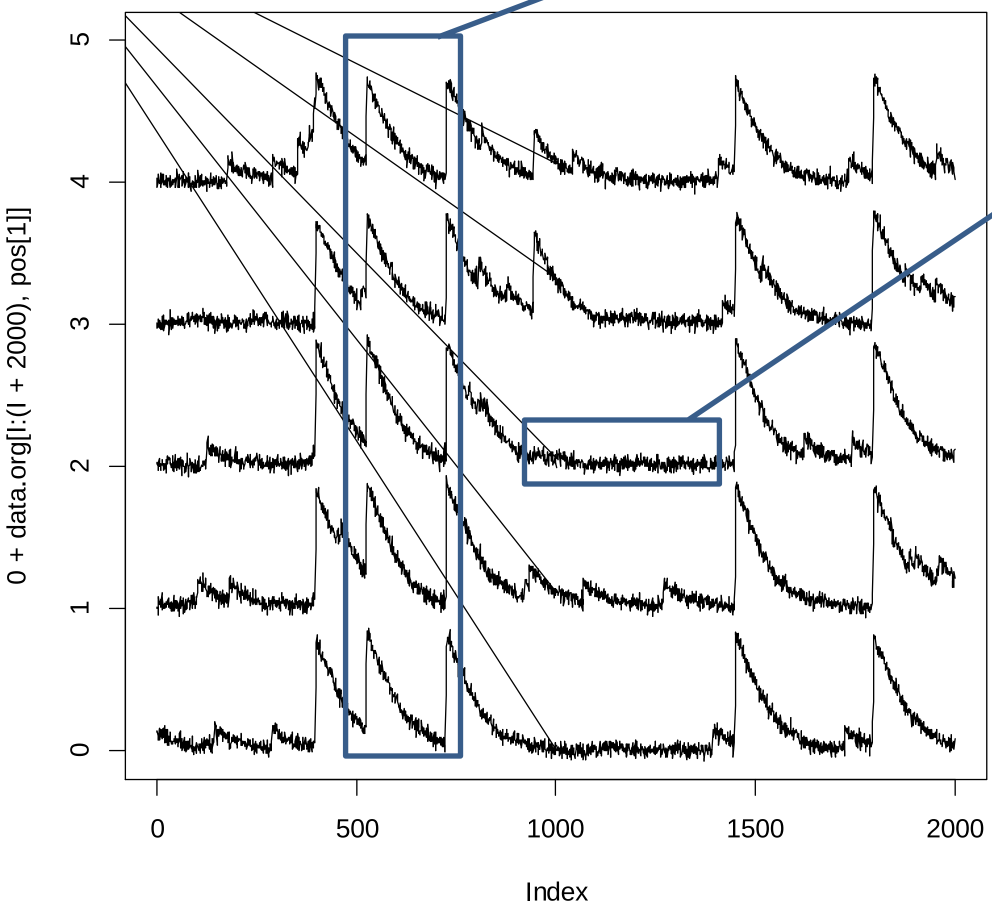


Dataset consists of:

- Several networks with 1000 neurons
- For each neuron, 1 hour time series of neural activity sampled at 20ms with values normalized in [0,1]
- Training networks contain connectivity labels

Evaluation based on the Area under the ROC Curve

What is hard about this challenge?



<u>Episodes of synchronous bursting:</u>

- Time periods conveying low connectivity information

<u>Background noise and light scattering artifacts:</u>

- Noise and signal can be confused

<u>Slow frame rate of ~20 ms is 10 times slower than neuron's firing dynamic:</u>

- Direct vs indirect effects and causal relationships are hard to detect

Solution guidelines

<u>Idea:</u>
“Optimizing a single connectivity indicator may be a limiting strategy because it will tend work optimally for particular conditions of synchronous bursting and noise.”

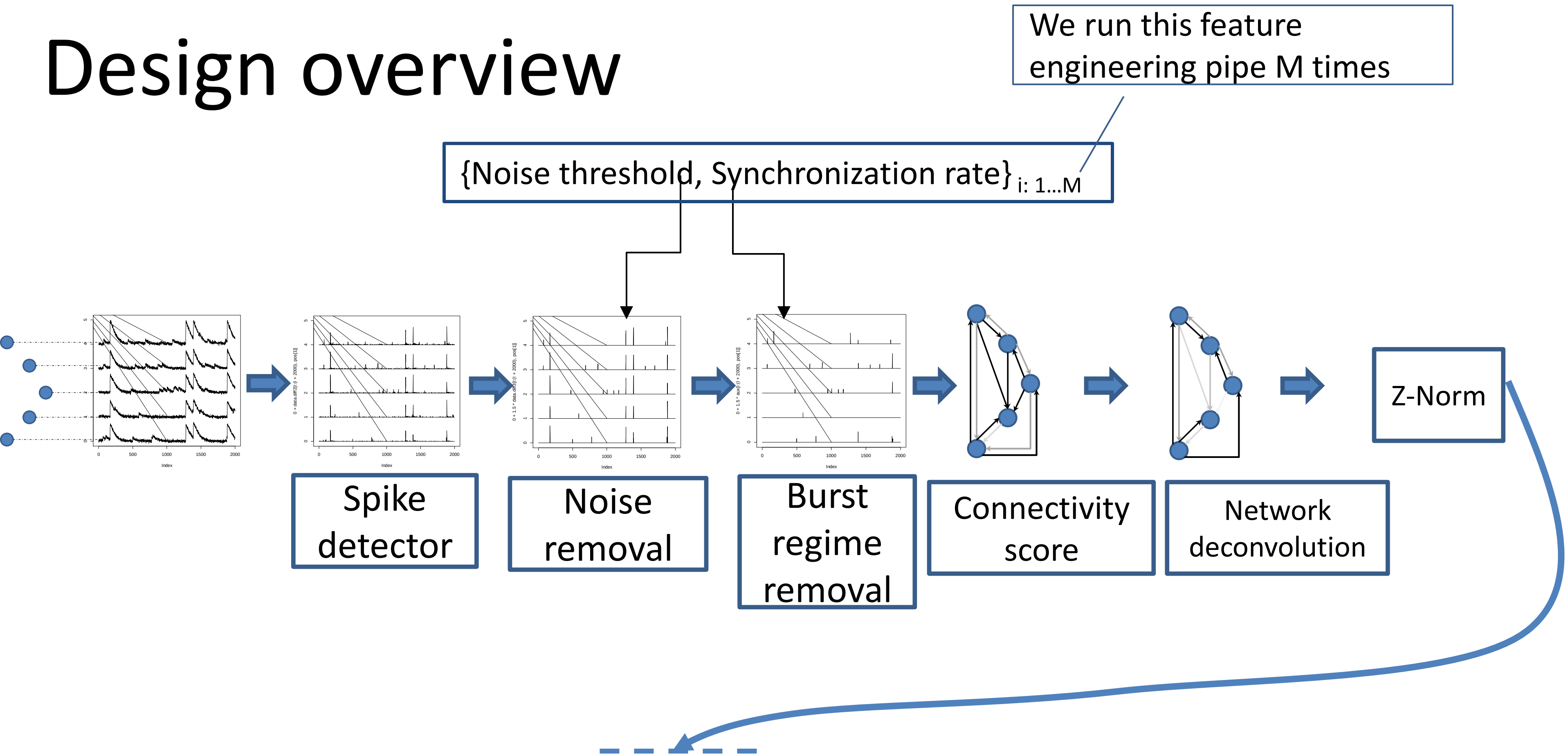
<u>How:</u>
To approximate a function able to combine several connectivity indicators optimized for different conditions of synchronous bursting and noise.

<u>Additional non-functional requirements:</u>

- **Given the large number of connectivity indicators that need to be computed (training and test networks, and for each neuron pair), the computational complexity of connectivity indicators must be low.**

- **Given the a priori large number of possible solutions to each of the main challenge problems (synchronous bursting, noise and direct vs indirect effects), we favored a modular approach.**

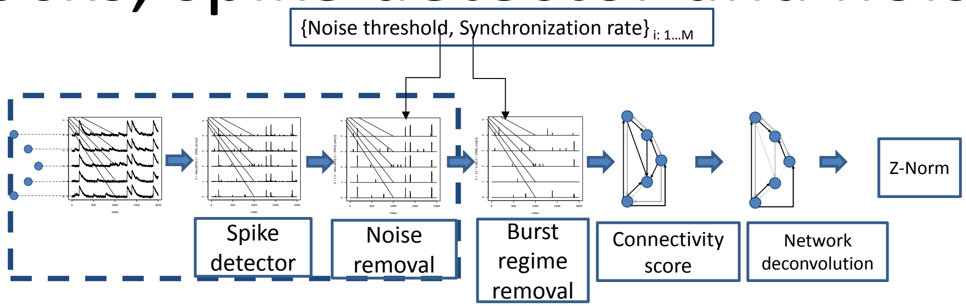
Design overview

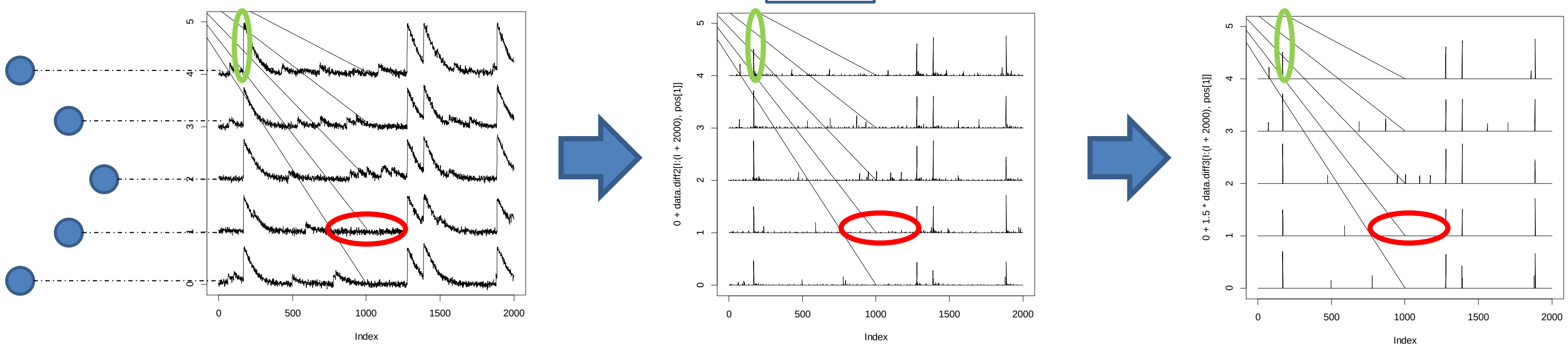


Neuron pair	$\{NT,SR\}_1$	$\{NT,SR\}_2$	$\{NT,SR\}_3$ …	$\{NT,SR\}_M$	True label
1 -> 1	0	0	0	0	0
1 -> 2	0.9	0.87	0.85	0.92	1
2 -> 1	0.9	0.87	0.85	0.92	1
1 -> 3 …	0.3	0.34	0.35	0.24	0
N -> N	0	0	0	0	0

F(NN time series, i, j) ϵ [0,1]

Building blocks, spike detector and noise removal





Goal: To infer the actual spike train of neuron i, given the time series of neural activity:

$$F_i = \{F_i^1, F_i^2, ..., F_i^T\} \rightarrow n = \{n_i^1, n_i^2, ..., n_i^T\}$$

How: Fast-oopsi developed by Joshua Vogelstein:

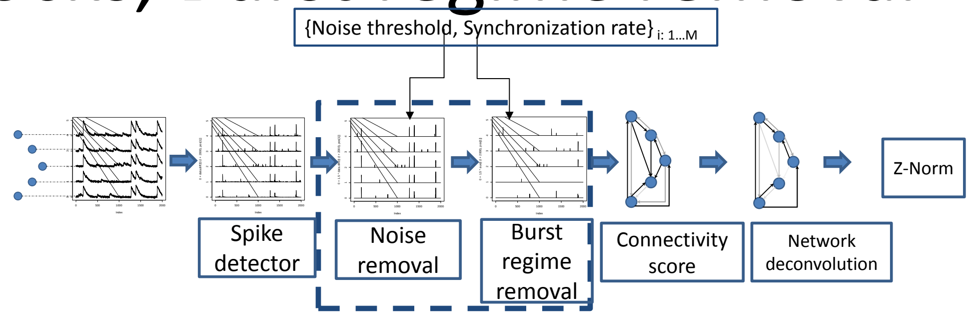
$$\hat{n} = arg\, max_n Pr(n|F)$$

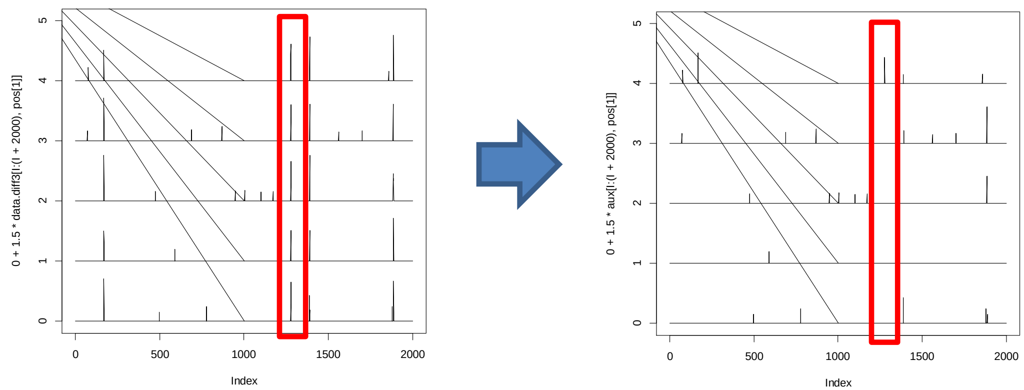
Post-processing: The *Noise Removal* Step removes spikes below a parameter value *Noise Level*

Evaluation vs Backward difference ($n_i^t = F_i^t - F_i^{t-1}$): using the complete feature engineering pipeline, the training network *Normal$_1$* and a large set of Noise Levels $\{NL_1,...,NL_M\}$:

Fast-oopsi	Difference
mean $\{AUC_{1...M}\}$=0.909 **max$\{AUC_{1...M}\}$=0.932**	mean $\{AUC_{1...M}\}$=0.902 max$\{AUC_{1...M}\}$=0.929

Building blocks, Burst regime removal





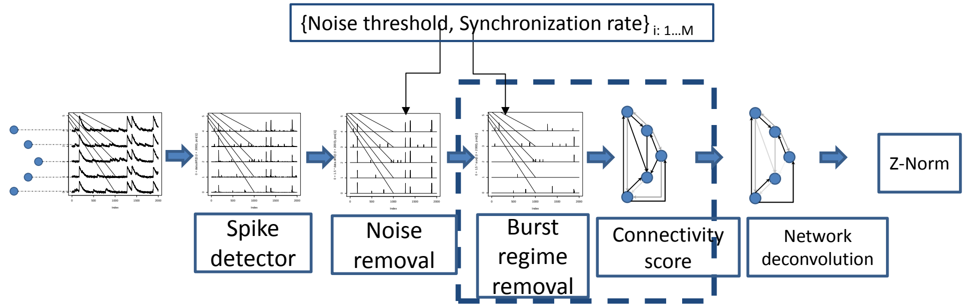
Goal: To remove episodes of synchronous bursting because they convey low connectivity information

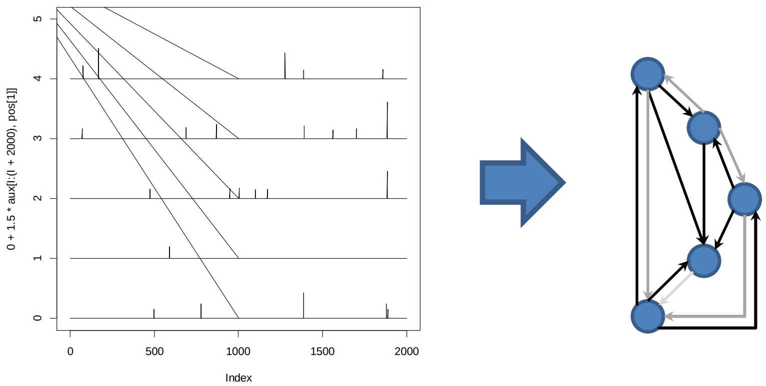
How: To remove time steps t in all neurons s.t.

$$\sum_{i=1}^{N} \mathbb{1}\{n_i^t \neq 0\} > SR.N$$

n_i^t: probability of neuron i spiking at time t
N: number of neurons
SR: Synchronization Rate or percentage of neurons allowed to fire at the same time t

Building blocks, Connectivity score





Goal: To compute a connectivity indicator between all neuron pairs given the spike trains computed in previous steps

Non-functional requirements: Computational performance should be low

2 training networks + test and validation networks

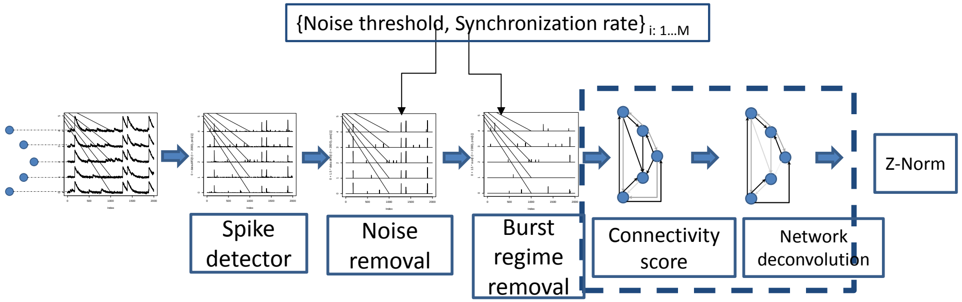
1000 neurons per network

M connectivity indicators for each neuron pair

How: Correlation shows good AUC with reasonable computational performance

Limitations: Unable to identify directed connections

Building blocks, Network deconvolution



<u>Goal:</u> To eliminate the combined effect of indirect paths of arbitrary length (Slow frame rate of 20 ms ~ 10 times slower than neuron's firing dynamic)

<u>How:</u> Network Deconvolution algorithm developed by Soheil Feizi et Al. 2013:

$$C_{dir} = C_{obs}(I + C_{obs})^{-1}$$

1) To normalize in the interval [-1,1] the connectivity score matrix
2) To decompose with SVD the matrix obtained in step 1
3) To compute the eigenvalues of the deconvolved matrix according to:

$$\lambda_i^d = \frac{\lambda_i}{\lambda_i + 1}$$

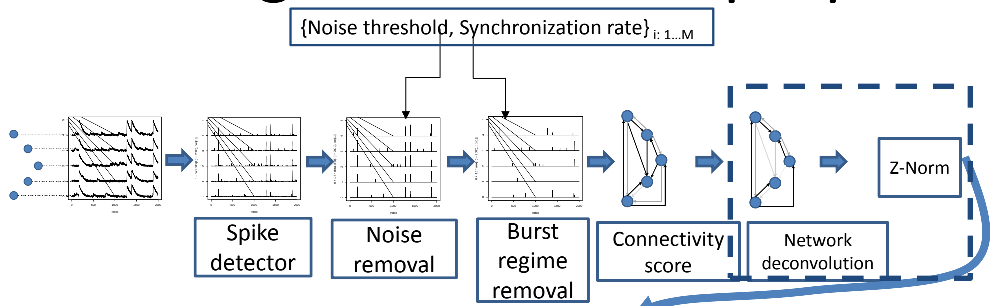
where λ_i being the i_{th} eigenvalue of the normalized matrix

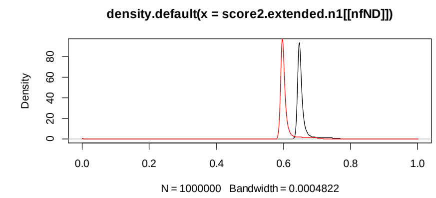
4) To compose the direct dependency matrix according to:

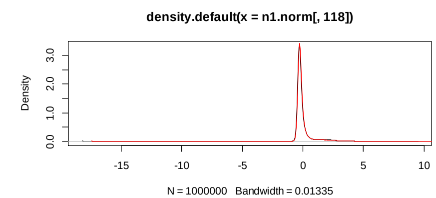
$$C_{dir} = UDU^{-1}$$

Where U is the matrix of eigenvectors and D is a diagonal matrix s.t. i_{th} diagonal element is λ_i^d

Building blocks, Training and test data preparation







Neuron pair	$\{NT,SR\}_1$	$\{NT,SR\}_2$	$\{NT,SR\}_3$ …	$\{NT,SR\}_M$	True label
1 -> 1	0	0	0	0	0
1 -> 2	0.9	0.87	0.85	0.92	1
2 -> 1	0.9	0.87	0.85	0.92	1
1 -> 3 …	0.3	0.34	0.35	0.24	0
N -> N	0	0	0	0	0

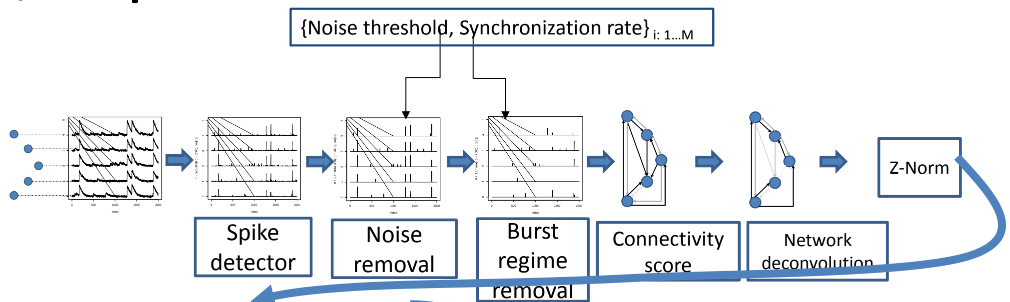
Goal:

1. To normalize the values of the deconvolved matrix so that networks using the same parameters (i.e. NL, SR) have similar distributions.
2. To create data tables to train and test a supervised model

How: For each network we create one table s.t.:

- Row: connectivity indicators for neuron pair (i,j) computed according to different Noise Levels and synchronization rates
- Column: connectivity indictors for all neuron pairs computed according to a particular Noise Level and Synchronization Rate
- Column-wise Z-normalization to normalize distributions across different networks
- Rows corresponding to self-loops, e.g. (i,i) are all 0 (We do not learn self-loops)
- We ignore directed connections, e.g. (i,j)=(j,i)

Building blocks, Supervised model



Test

Neuron pair	$\{NT,SR\}_1$	$\{NT,SR\}_2$	$\{NT,SR\}_3$ ...	$\{NT,SR\}_M$	True label
1 -> 1	0	0	0	0	?
1 -> 2	0.9	0.87	0.85	0.92	?
2 -> 1	0.9	0.87	0.85	0.92	?
1 -> 3 ...	0.3	0.34	0.35	0.24	?
N -> N	0	0	0	0	?

Normal$_1$

Neuron pair	$\{NT,SR\}_1$	$\{NT,SR\}_2$	$\{NT,SR\}_3$ ...	$\{NT,SR\}_M$	True label
1 -> 1	0	0	0	0	0
1 -> 2	0.9	0.87	0.85	0.92	1
2 -> 1	0.9	0.87	0.85	0.92	1
					0
					0

Normal$_2$

Neuron pair	$\{NT,SR\}_1$	$\{NT,SR\}_2$	$\{NT,SR\}_3$ ...	$\{NT,SR\}_M$	True label
1 -> 1	0	0	0	0	0
1 -> 2	0.92	0.37	075	0.62	0
2 -> 1	0.9	0.77	0.75	0.92	1
1 -> 3 ...	0.34	0.44	0.55	0.34	0
N -> N	0	0	0	0	0

Random Forest$_1$ (R_1)
Gradient Boosting Machine$_1$ (G_1)
Random Forest$_2$ (R_2)
Gradient Boosting Machine$_2$ (G_2)

$P=(R_{1+}G_1+R_{2+}G_2)/4$

Training sub-sampling strategy:

- Self-loop rows are not included
- All rows with label 1 are included
- 5% of rows with label 0 are included

Experiments

Model

Test network

AUC	GTE	Feature	R_1	G_1	R_2	G_2	R_3	G_3	R_1+G_1	R_2+G_2	R_3+G_3	$R_i+G_i+R_j+G_j$	$R_1+G_1+R_2+G_2$
$Normal_1$	.885	.932	-	-	.9392	.9388	.9394	.9386	-	.9398	.9398	.9401	
$Normal_2$	.889	.93	.9399	.9401	-	-	.9403	.9399	.9409	-	.9409	.9413	
$Normal_3$	.884	.933	.9393	.9396	.9392	.9396	-	-	.9402	.9402	-	.9405	
Test	.893												0.94063

Best individual connectivity indicator is better than GTE

Simple supervised model is better than best individual conn. indicator

We improve performance an reduce variance by averaging predictions from many models

Conclusions, limitations and future work

Conclusions

- Third best model according to the private leadeboard: AUC_{GTE}=.893, AUC_{best}=.94063
- To approximate a function able to combine several connectivity indicators optimized for different conditions of synchronous bursting and noise seems to be a good idea

Limitations

- We are unable to identify the connectivity direction and self-loops
- High computational cost due to the computation of large number of connectivity indicators (e.g. training our last model took > 48 hours on i7 laptop with 32 Gb)

Future work:

- Add few directional connectivity indicators to the predictive model (e.g. GTE)
- Increase the number of features by using a finer grid of parameters (i.e. $\{\text{Noise threshold, Synchronization rate}\}_{i:\,1\ldots M}$)
- Increase the performance of predictive models by using semi-supervised variants of random forest and gradient boosting machines