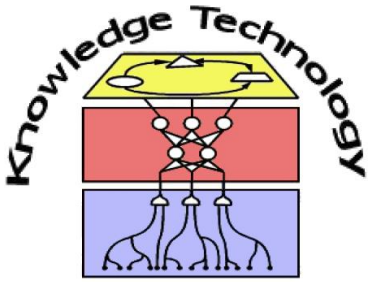
Neural Networks

Lecture 6: Echo State Networks



http://www.informatik.uni-hamburg.de/WTM/

Dr. Xavier Hinaut

Inria
Bordeaux, France

Team website:
team.inria.fr/mnemosyne/fr

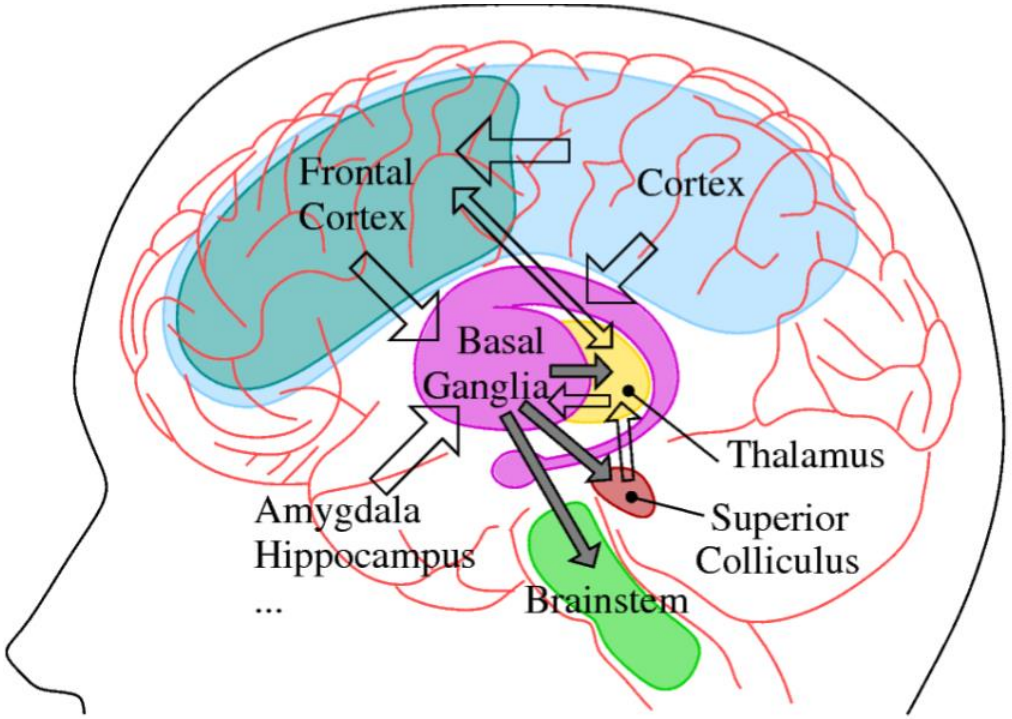
twitter: @neuronalX
email: xavier.hinaut@inria.fr
github.com/neuronalX/EchoRob

Computational Neuroscience Cognition Memory Brain & Body Complex systems

Mnemosyne

informatics mathematics Inria

Model the brain as a synergy of memories



imn institut des maladies neurodégénératives

cnrs

LaBRI

Initiative d'Excellence université de Bordeaux

université de BORDEAUX

IPB INSTITUT POLYTECHNIQUE BORDEAUX

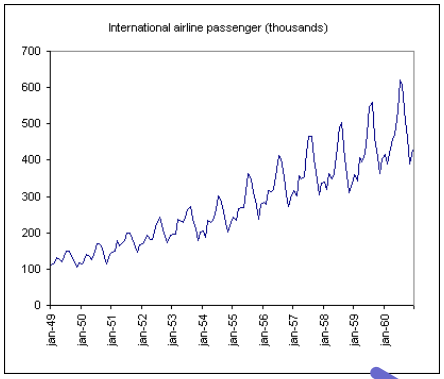
Motivation

- 1: Why to process temporal series/sequences?
  - Find structure in time, encode the context, ... (e.g. Language)
  - Recognize gestures (Doreen), human actions (Luiza), …
  - Make predictions on future time steps (e.g. Stock market)
  - Animals/Robots need to learn/process streams efficiently
- 2: Why using Recurrent Neural Network?
  - Very powerful to encode non-linear contextual information
  - + or - biologically plausible (depending on algorithm used)
  - Problem: very costly and long to train for big networks ...

Motivation

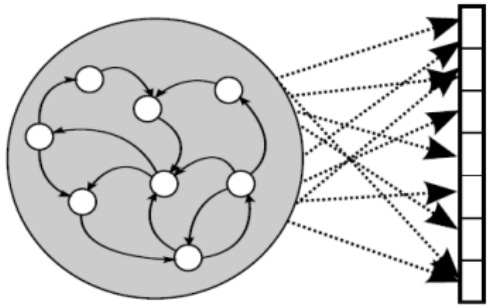
- 1: Why to process temporal series/sequences?
  - Find structure in time, encode the context, ... (e.g. Language)
  - Recognize gestures (Doreen), human actions (Luiza), …
  - Make predictions on future time steps (e.g. Stock market)
  - Animals/Robots need to learn/process streams efficiently
- 2: Why using Recurrent Neural Network?
  - Very powerful to encode non-linear contextual information
  - + or - biologically plausible (depending on algorithm used)
  - Problem: very costly and long to train for big networks ...
- 3: How to speed up the (supervised) learning?
- 4: How the brain processes sequences (and language)?
- 5: How to not unfold time while learning/processing data?

Do not unfold time, please!



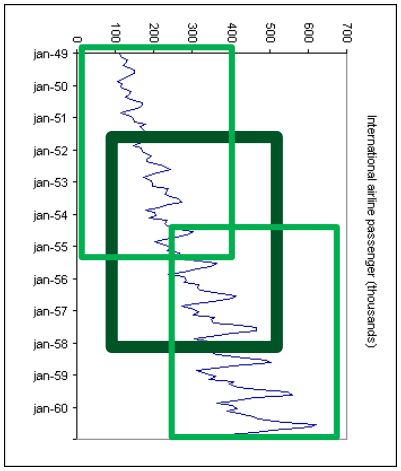
Time

Processing



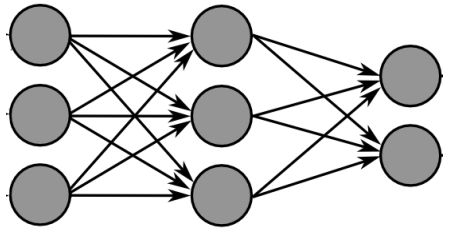
Recurrent Neural Net.

Time



Sliding window

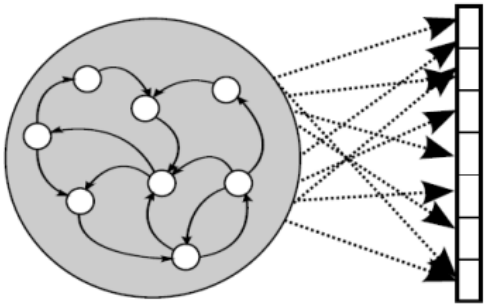
Processing

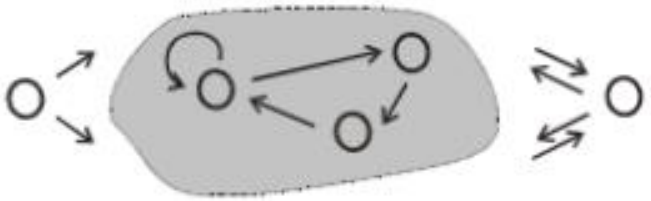


Feed Forward Neural Net.

Do not unfold time, please!

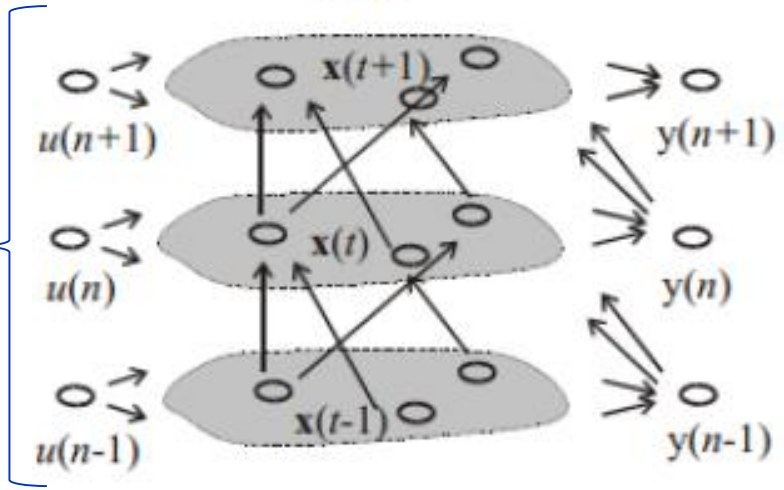
Still, back-propagation can be used with RNNs...





Unfold time

Duplicate the network for each time step.

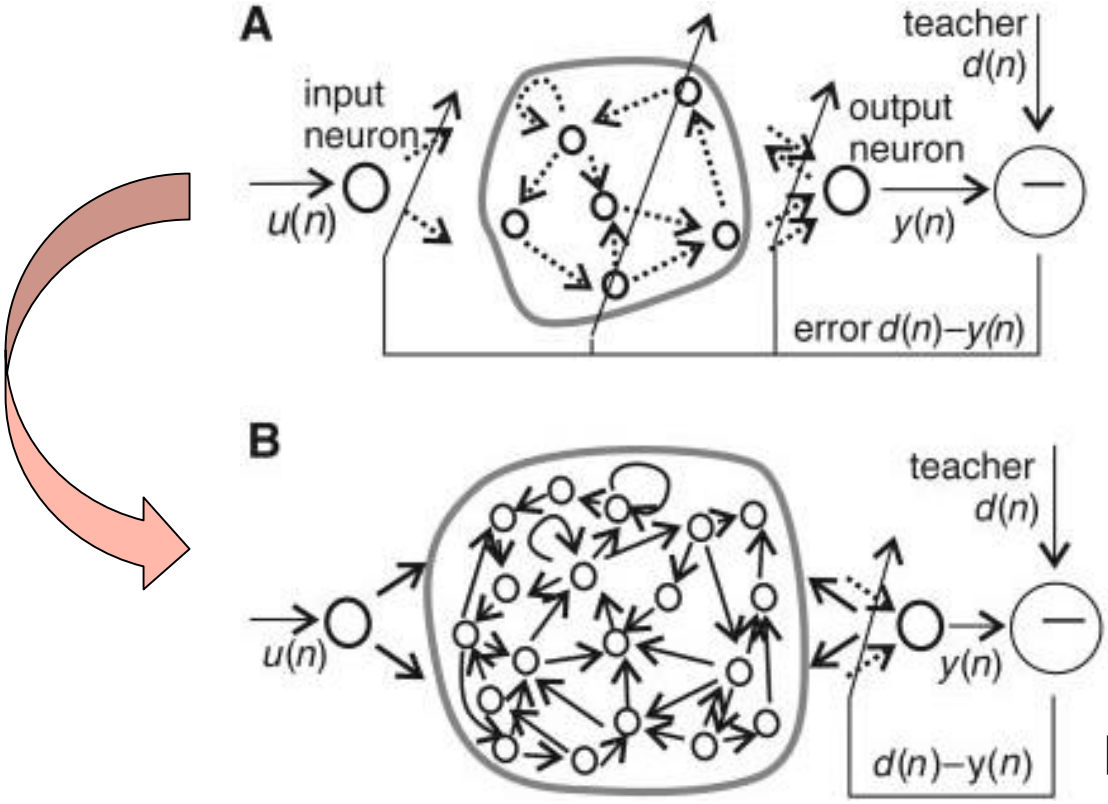


Outline

- **1: Introduction on Echo State Networks (ESN)**
  - How to speed up learning with RNNs?
  - Why should it work?
- 2. Some examples
- 3. How does it work? Diving in the reservoir
- 4. Initialization and parameters
- 5. Different types of Reservoir Computing
- 6. Practical advices for parameter optimization
- 7. Summary on ESN
- 8. Application to language processing
- 9. Conceptors: new way to train Reservoirs

How to speed up the (supervised) learning?

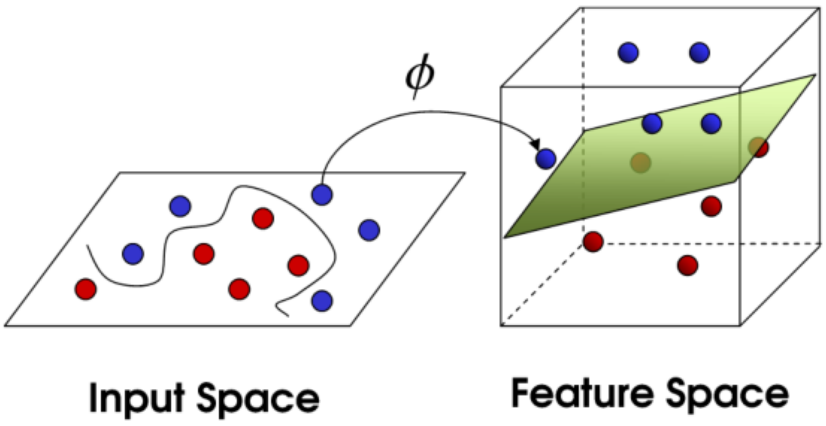
- Do not train all the network (e.g. ***just the output layer***)
- ***Randomize*** untrained connections (input & hidden layers)
- Use ***linear*** methods for training (e.g. Linear regression)

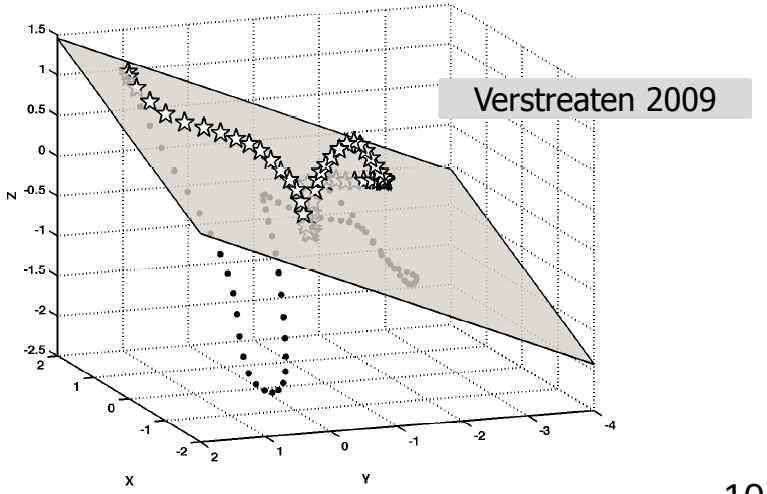


[Jaeger & Hass 2004]

Why should it work?

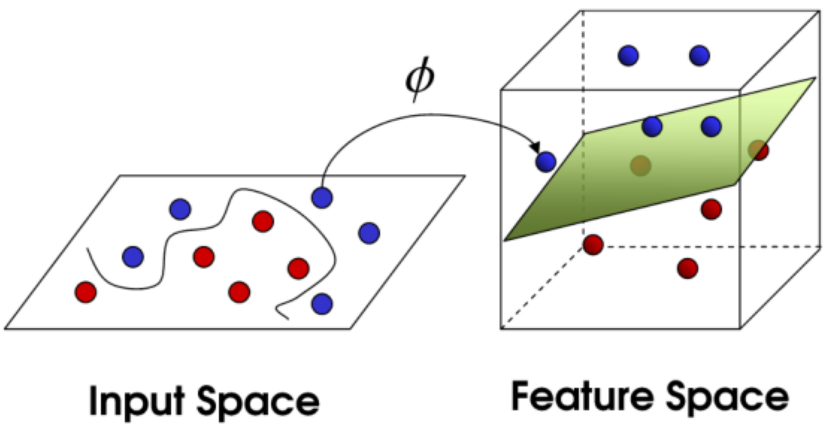
- Big hidden layer (100~1000 units)
  - Projection of the inputs in a ***high-dimensional space***
  - A lot of ***non-linear computations*** are performed based on the inputs
  - ***Rich temporal dynamics*** (even if random static network)
  - Reservoir = (big) hidden layer of the ESN

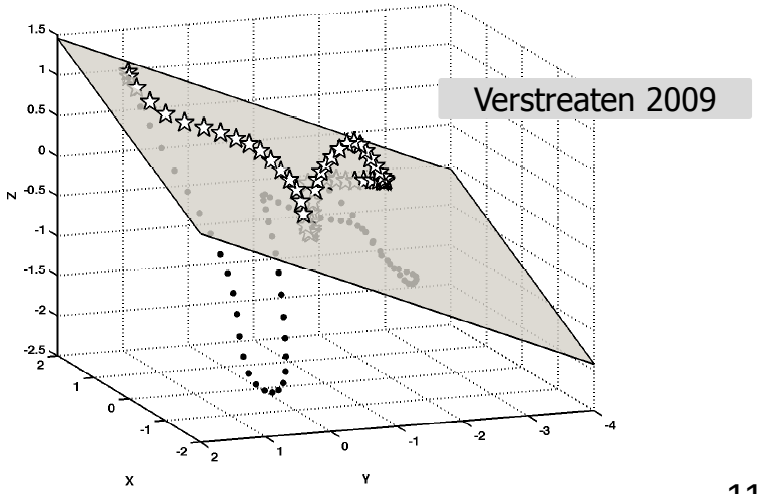




Why should it work?

- Linear method to ***extract only useful information***
  - From all the information present in the hidden layer (i.e. the reservoir) just pick what is needed for the output
  - No need to use non-linear learning method
    - Non-linear computations are already performed in the reservoir
- Like a ***temporal SVM*** (Support Vector Machine)

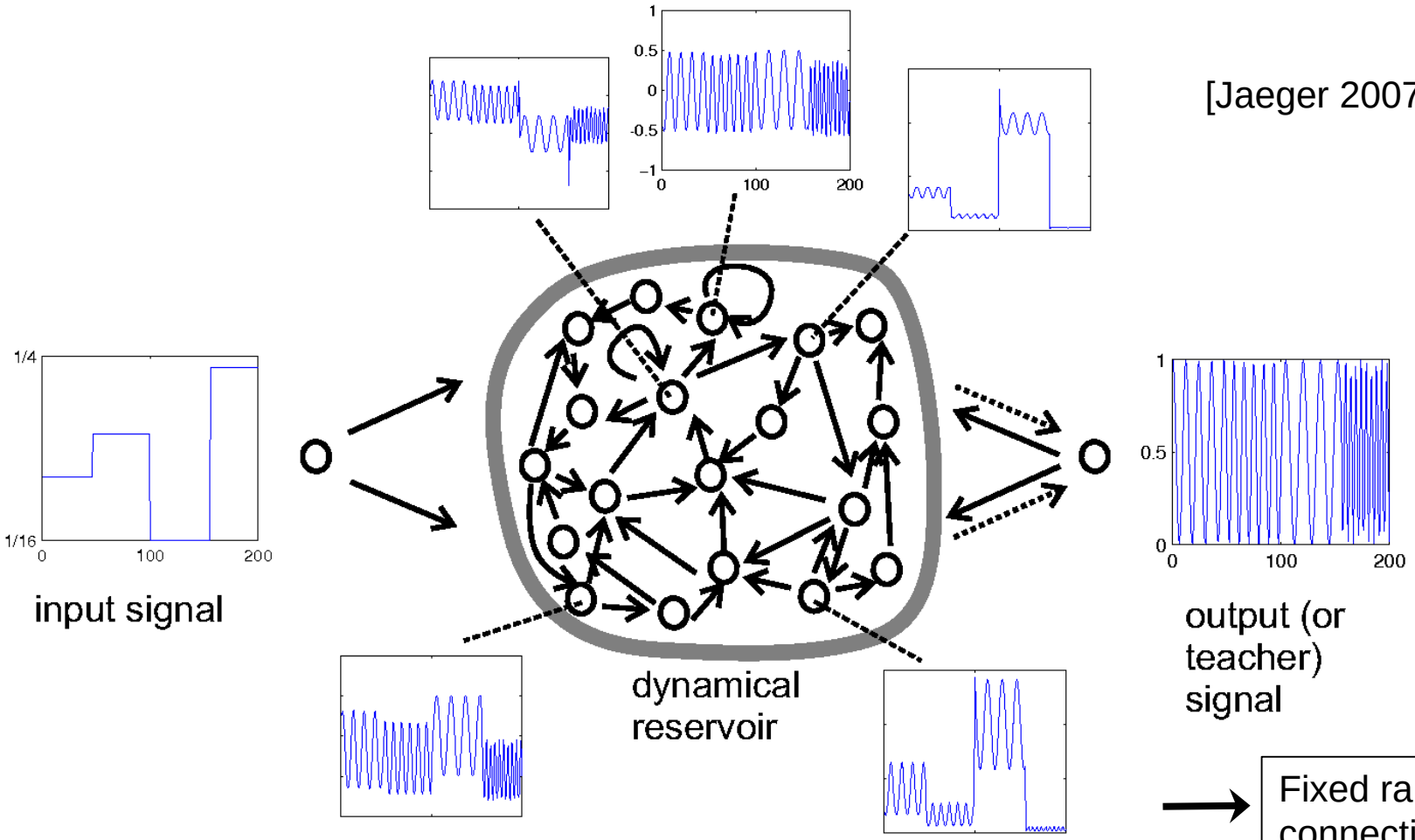




Outline

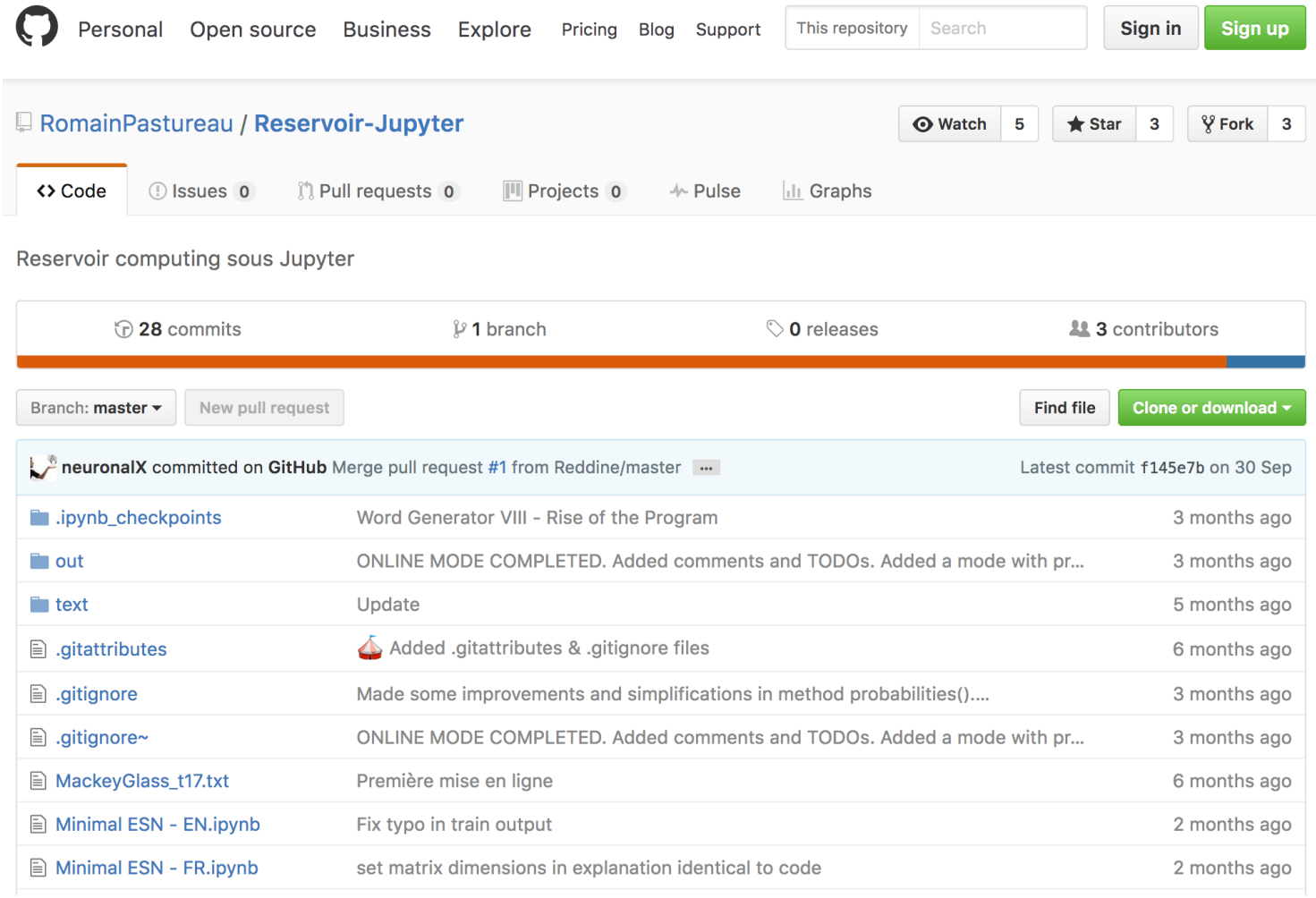
- 1: Introduction on Echo State Networks (ESN)
- **2. Some examples**
  - Tunable frequency generator
  - Chaotic time-series prediction
- 3. How does it work? Diving in the reservoir
- 4. Initialization and parameters
- 5. Different types of Reservoir Computing
- 6. Practical advices for parameter optimization
- 7. Summary on ESN
- 8. Application to language processing
- 9. Conceptors: new way to train Reservoirs

Example 1: Tuneable frequency generator



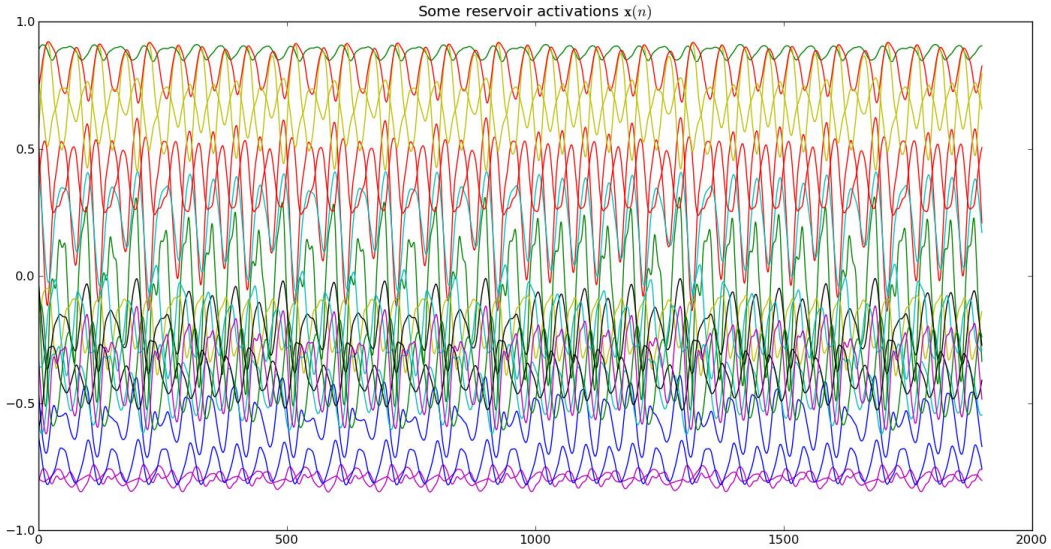
- Basic schema of an ESN

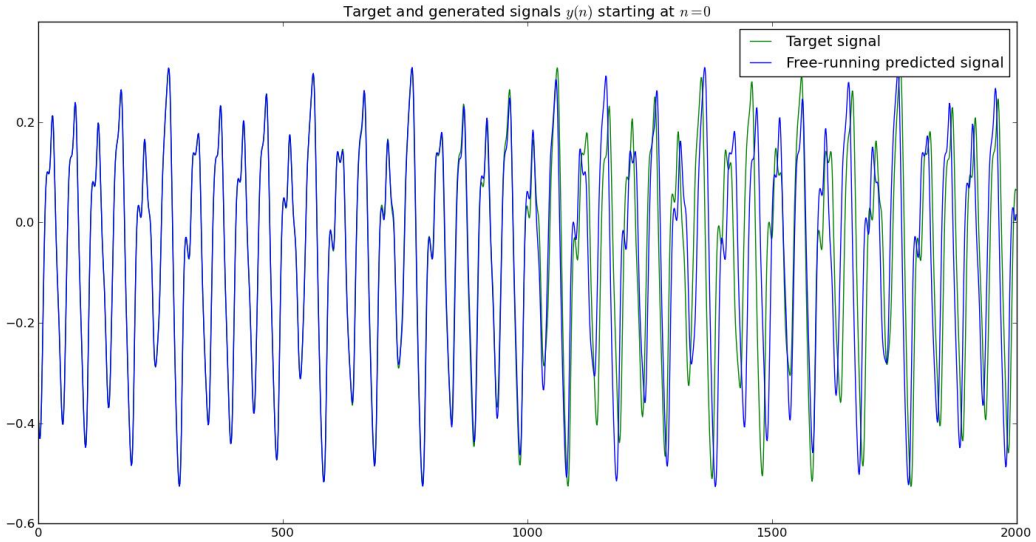
Tutorial with jupyter notebook



https://github.com/neuronalX/Reservoir-Jupyter

Example 2: Chaotic time-series prediction



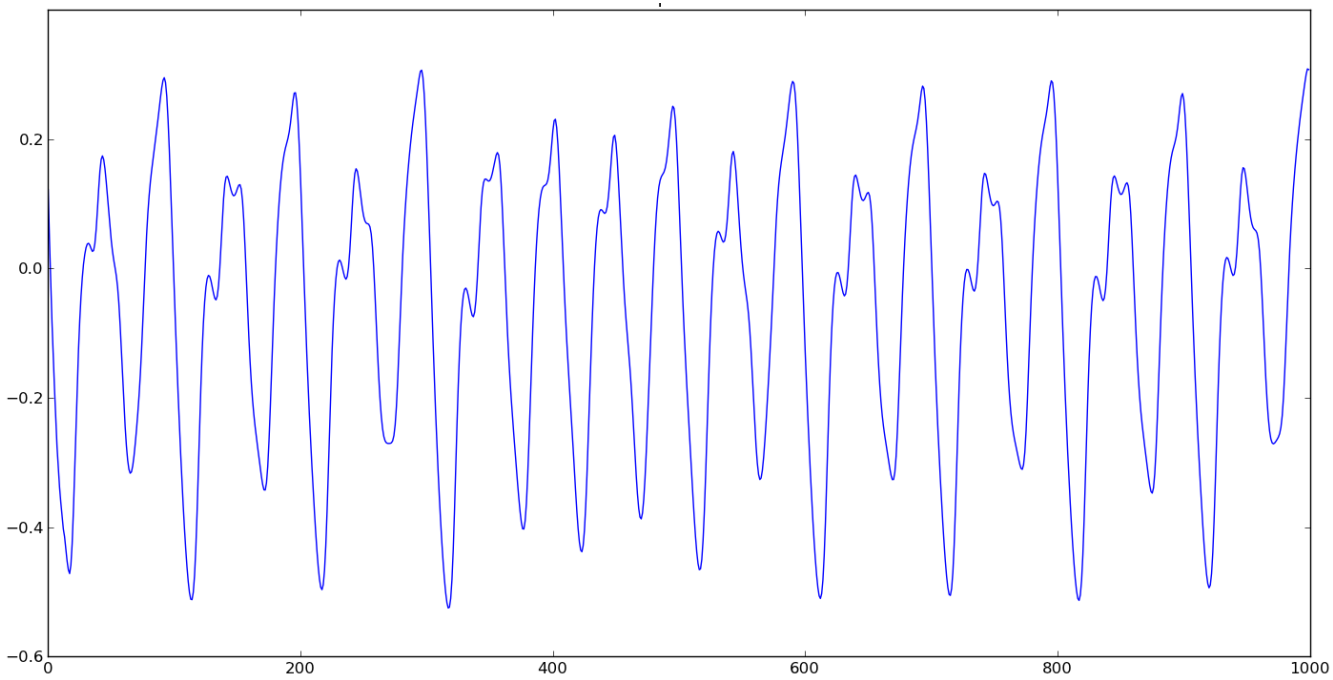


- Activity of 20 units inside the reservoir for 2000 time steps

- Output activities (teacher and real)

[Lukoševičius, minimalESN.py]

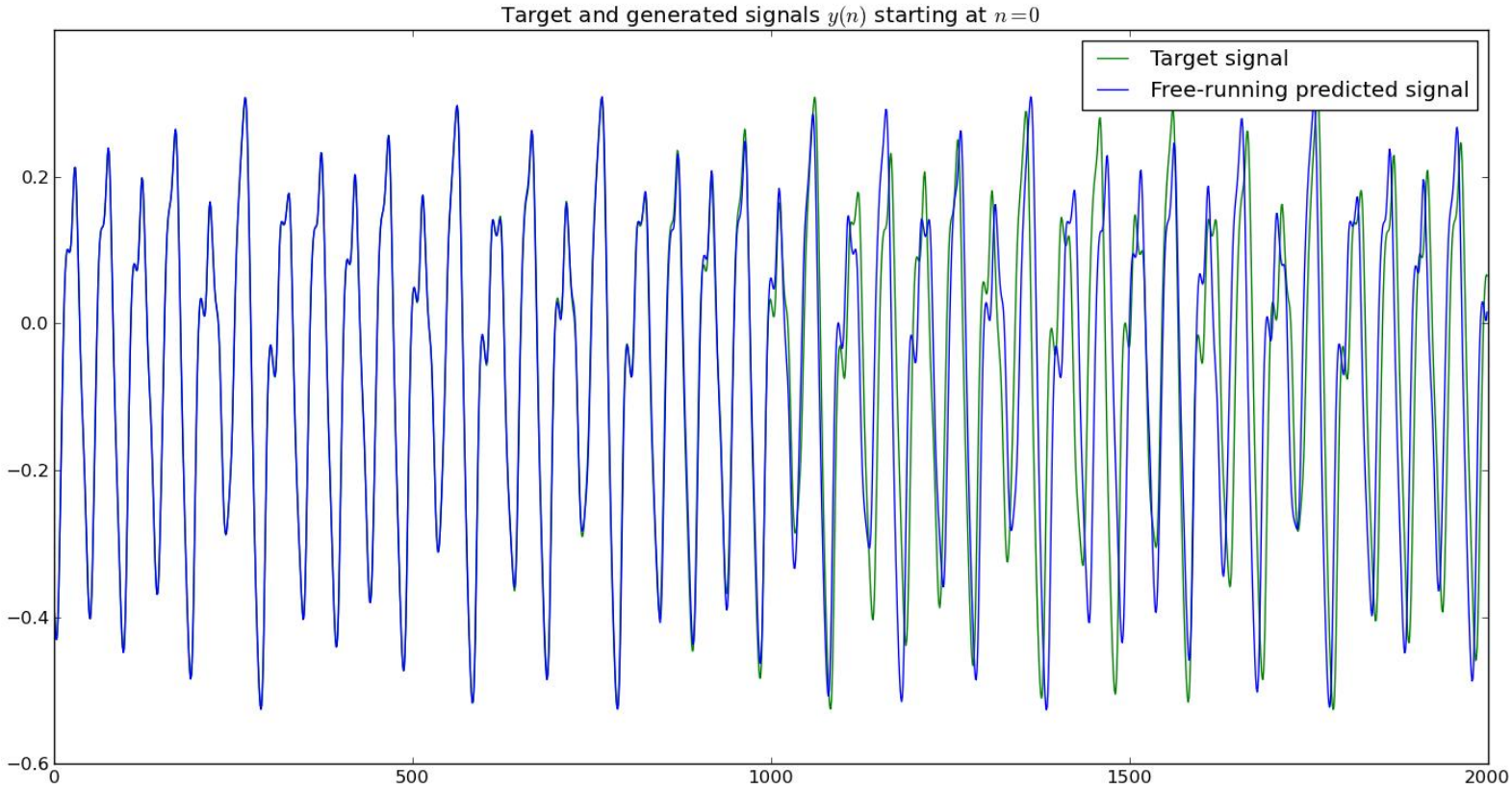
Example 2: Chaotic time-series prediction



- Task: prediction of the Mackey-Glass chaotic time-series with delay parameter (τ = 17)
- Based on time step ***n***, the network has to give the next time ***n+1***

[Lukoševičius, minimalESN.py]

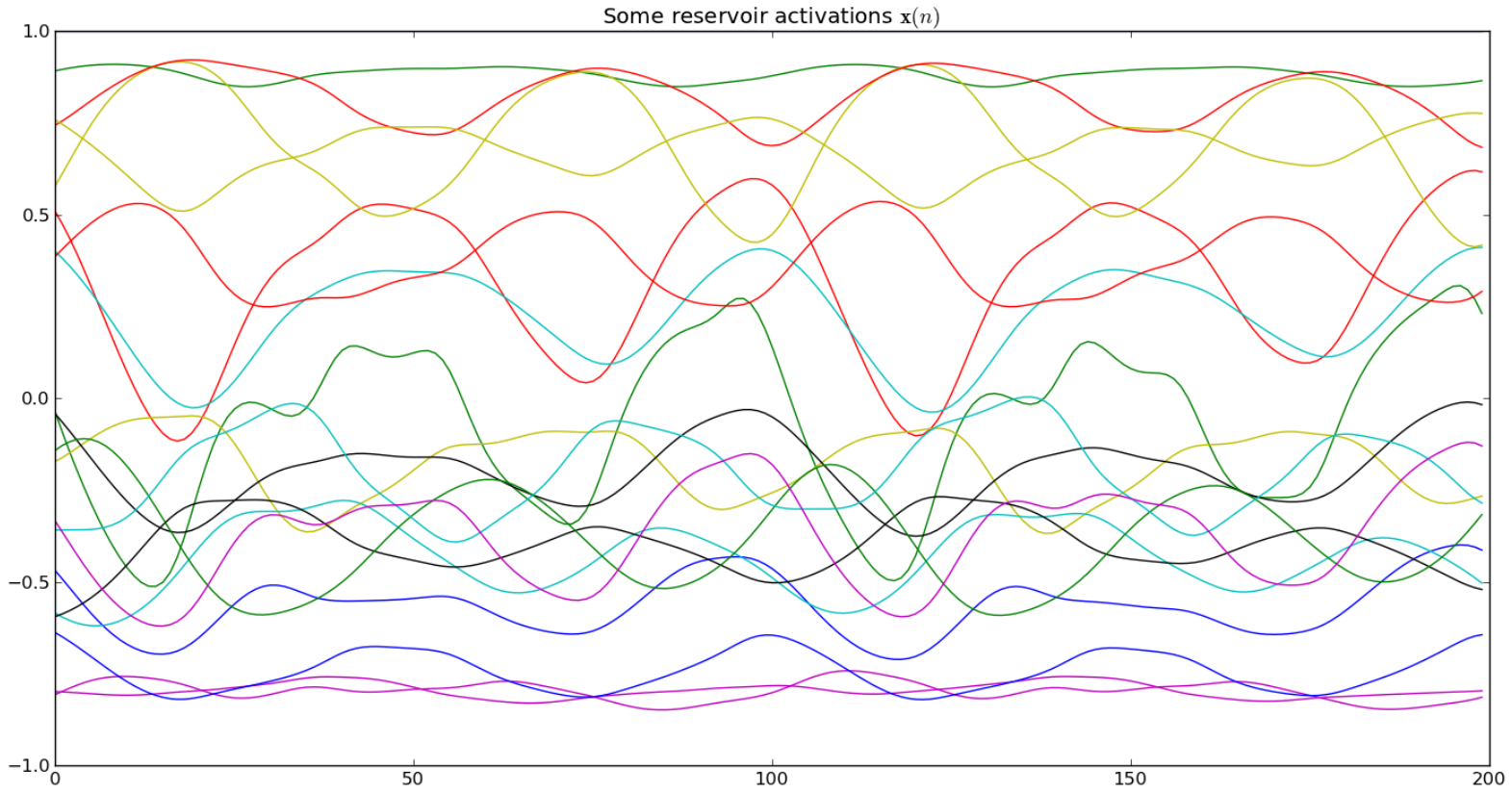
Example 2: Chaotic time-series prediction



- Green: Target signal (correct output)
- Blue: Predicted signal (real output)

[Lukoševičius, minimalESN.py]

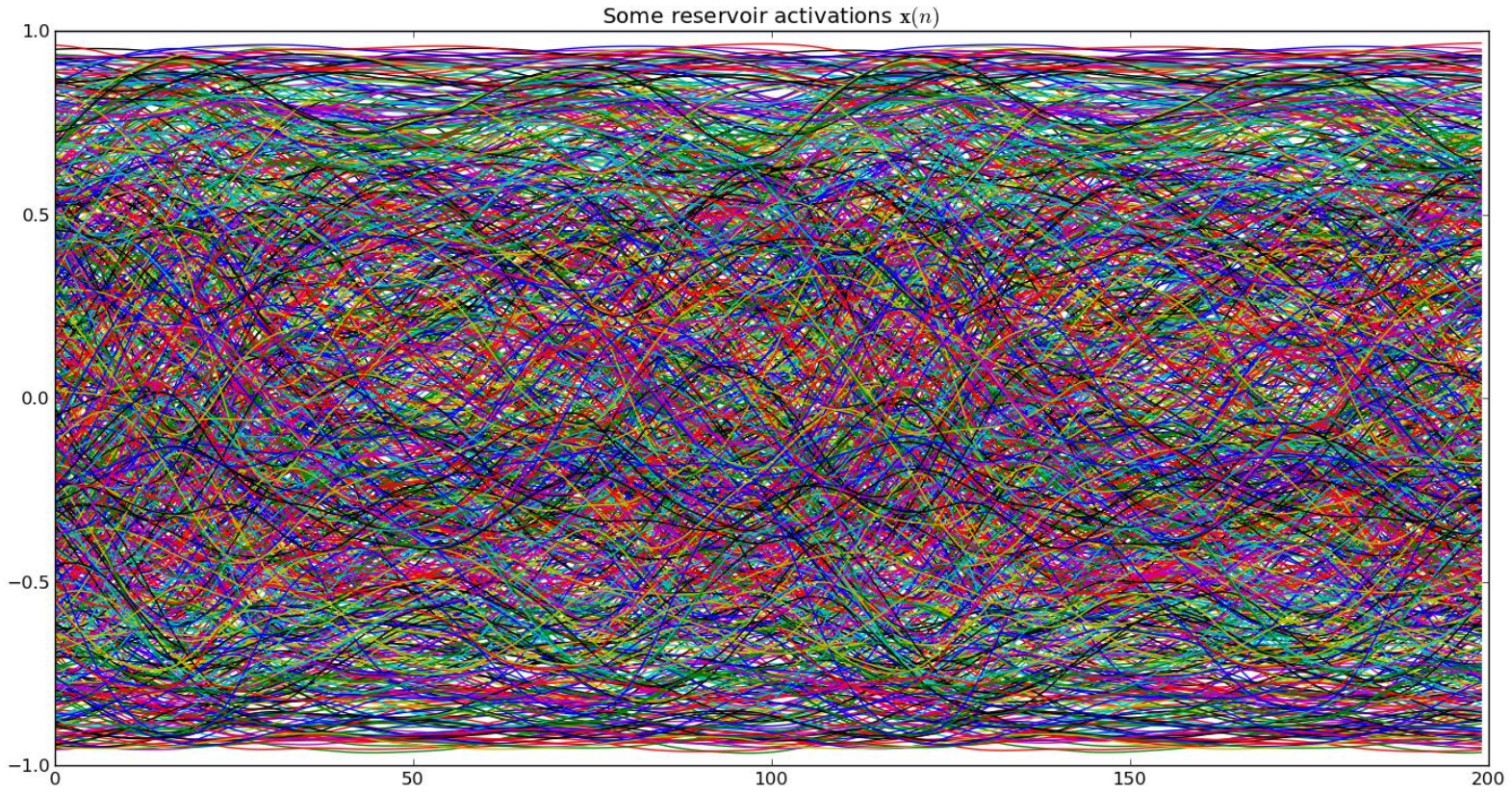
Example 2: Chaotic time-series prediction



- Activity of 20 units inside the reservoir for 200 time steps

[Lukoševičius, minimalESN.py]

Example 2: Chaotic time-series prediction



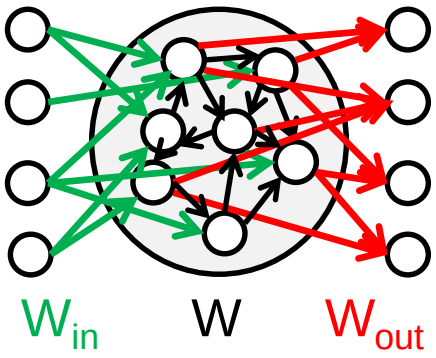
- Activity for all 1000 units inside the reservoir for 200 time steps

[Lukoševičius, minimalESN.py]

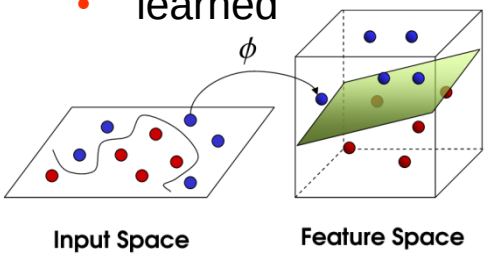
Outline

- 1: Introduction on Echo State Networks (ESN)
- 2. Some examples
- **3. How does it work? Diving in the reservoir**
- 4. Initialization and parameters
- 5. Different types of Reservoir Computing
- 6. Practical advices for parameter optimization
- 7. Summary on ESN
- 8. Application to language processing
- 9. Conceptors: new way to train Reservoirs

How does it work? Diving in the Reservoir



- W_{in}, W:
  - randomly generated
- W_{out} :
  - learned



Jaeger 2001, Jaeger et al. 2007, Lukosevicius 2012

Reservoir states update for *leaky* integrator neurons

$$\mathbf{x}(t) = (1 - \frac{1}{\tau})\mathbf{x}(t-1) + \frac{1}{\tau} f(\mathbf{W}^{in} u(t) + \mathbf{W}\mathbf{x}(t-1)) \quad (1)$$

- x the reservoir state vector,
- u the input vector,
- W the reservoir recurrent weight matrix,
- W^{in} the weight matrix from input layer to the reservoir,
- τ the time constant of a reservoir unit,
- f the activation function (i.e. *tanh*) of a reservoir unit.

Output layer (read-out) state update

$$\mathbf{y}(t) = \mathbf{W}^{out}\mathbf{x}(t) \quad (2)$$

Offline learning of output weights

$$\mathbf{W}^{out} = Y^d Z^+ \quad (3)$$

- Y^d the concatenation of the desired outputs,
- Z the concatenation of reservoir states, with $Z = [X; 1]$
- $^+$ denotes the pseudo-inverse.

Offline Learning

- Offline learning with linear regression:

$$W_{out} = Y_{teacher} \, (Z^T)^+$$

  - U, X, $Y_{teacher}$ concatenation of all inputs *u(n),* reservoir states *x(n)* and desired outputs $y_{teacher}(n)$ *resp.*
  - W_{out}: output weight matrix (defined by learning proc.)
  - Z = [1;U;X] (concatenation of all states for all time steps)
  - Z^T: transpose of Z
  - A^+: pseudo-inverse of A

- Use pseudo-inverse for stability issues
  - Works even if Z is ill-conditioned (~ not invertible)

Offline Learning with regularization

- Prevent overfitting with ridge regression:

$$W_{out} = Y_{teacher} \, Z^T (ZZ^T + \beta I)^{-1}$$

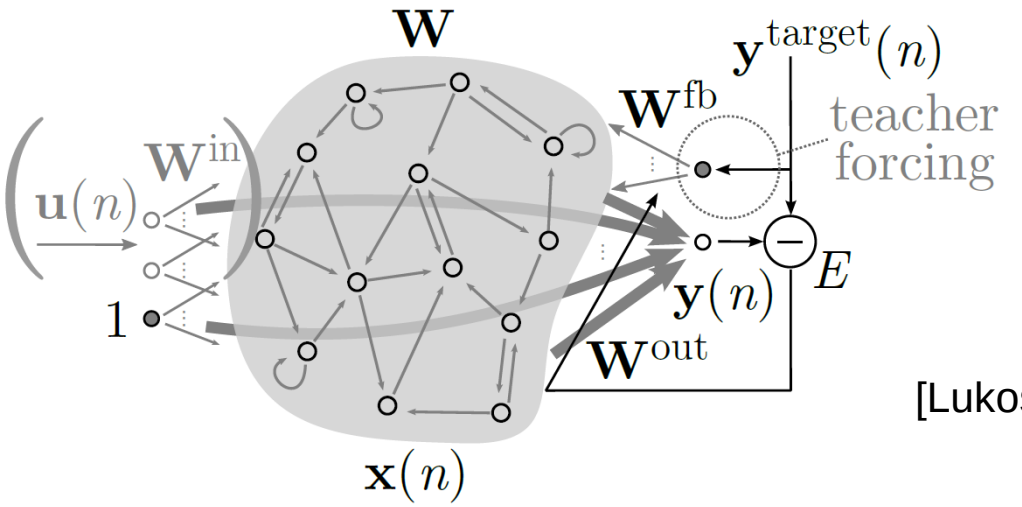
  - U, X, $Y_{teacher}$ concatenation of all inputs *u(n),* reservoir states *x(n)* and desired outputs $y_{teacher}(n)$ *resp.*
  - W_{out}: output weight matrix (defined by learning proc.)
  - Z = [1;U;X] (concatenation of all states for all time steps)
  - Z^T: transpose of Z
  - A^{-1}: inverse of A
  - β: ridge parameter

- Equivalent to add noise in reservoir (and inputs) states

With feedback from outputs to reservoir

- General reservoir update equation (non-leaky):

$$x(n) = f\left(Wx(n-1) + W_{in}[1; u(n)] + W_{fb}\, y(n-1)\right)$$



[Lukosevicius 2012]

- *x(n)*: reservoir state at time *n*
- *u(n):* input at time *n*
- *y(n): output at time n*
- *W*: reservoir weight matrix
- W_{in} *:* input weight matrix
- W_{fb} *:* feedback weight matrix

Outline

- 1: Introduction on Echo State Networks (ESN)
- 2. Some examples
- 3. How does it work? Diving in the reservoir
- **4. Initialization and parameters**
- 5. Different types of Reservoir Computing
- 6. Practical advices for parameter optimization
- 7. Summary on ESN
- 8. Application to language processing
- 9. Conceptors: new way to train Reservoirs

Initialization and parameters

- Generate random matrices W_{in} and W
  - With a centered distribution (gaussian, uniform, bimodal, …)
  - With sparse connectivity (5 to 30% of non-zero connections)
  - Apply *input scaling* to input weight matrix W_{in} :
    - $W_{in} \leftarrow W_{in}$ * input_scaling
  - Apply *spectral* radius (SR) to reservoir weight matrix:
    - $W \leftarrow W * SR_{defined} / SR_{real}$
- **Input scaling**:
  - How much the ***input drives*** the reservoir
- **Spectral radius** (SR):
  - "Excitability" of the ***reservoir dynamics***
  - Absolute of maximal eigenvalue of reservoir weight matrix *W*
  - Could be interpreted as the « temperature » of the dynamics

Initialization and parameters

- Leak rate α
  - Corresponds to the ***inverse of a time constant***
  - Could be interpreted as the inverse of the "mass" of the reservoir units

- Ridge parameter β
  - Optimal parameter dependent on reservoir instance
    - → but could be set globally for practical purposes
  - Ex: in [Lukoševičius, minimalESN.py]: β set to 10^{-8}

Initialization and parameters

- Spectral radius (SR) and *Echo State Property* (ESP):
  - ESP: ensure that SR is strictly inferior to 1 in order to have a system that is able will forget its initial state
    - ~ the system will not amplify very small perturbations
  - For a linear reservoir not leaky (leak rate $\alpha=1$)
    - SR<1: the system is not chaotic ; the activity will die out for zero input ; the system will forget its initial state
  - For non-linear reservoir (general case):
    - ***Use the ESP as a "soft" guide rule***
    - If leaky reservoir: you can explore values of SR far from 1

Outline

- 1: Introduction on Echo State Networks (ESN)
- 2. Some examples
- 3. How does it work? Diving in the reservoir
- 4. Initialization and parameters
- **5. Different types of Reservoir Computing**
- 6. Practical advices for parameter optimization
- 7. Summary on ESN
- 8. Application to language processing
- 9. Conceptors: new way to train Reservoirs

Different types of Reservoir Computing

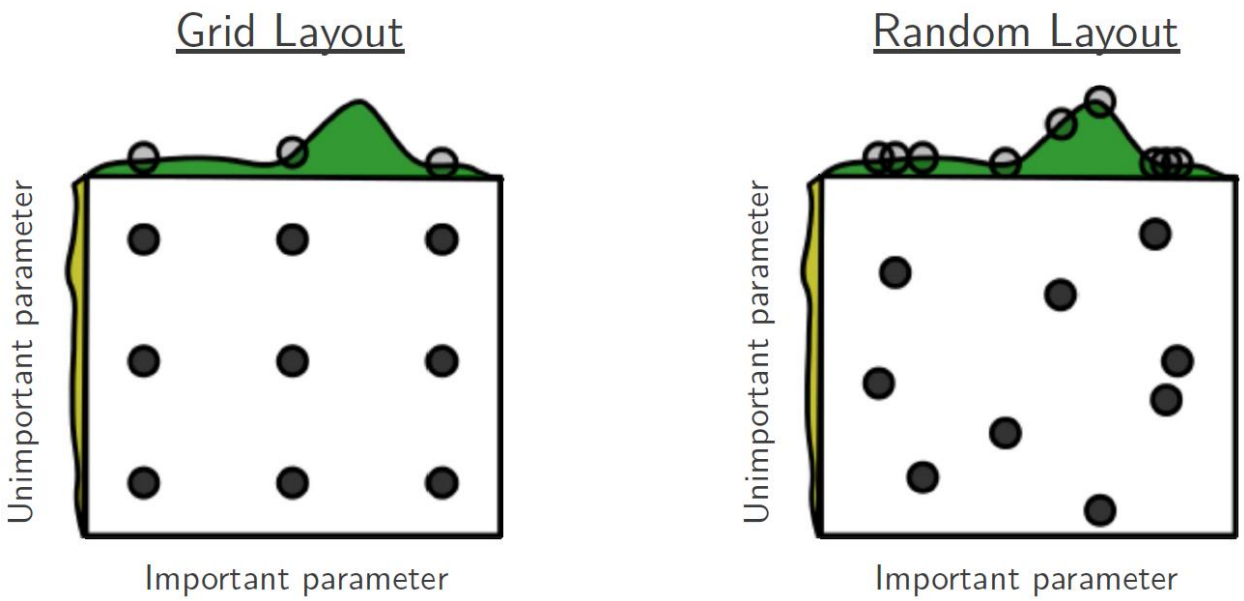
- ***Echo States Networks*** (H. Jaeger 2001)
  - Average firing-rate neurons (leaky or not)
- ***Liquid State Machines*** (W. Maass 2002)
  - ***Spiking*** neurons
  - With or without unsupervised plasticity inside reservoir
- Online learning:
  - "FORCE" learning (Sussillo & Abbott 2009)
  - Biologically plausible (reinforcement-like learning):
    - With exploratory Hebbian learning rule (Hoezer et al. 2012)
- First instances came from Computational Neuroscience:
  - [Dominey 1995]; [Buonomano & Merzenich 1995]
- Conceptors … [Jaeger 2014, 2017]

Outline

- 1: Introduction on Echo State Networks (ESN)
- 2. Some examples
- 3. How does it work? Diving in the reservoir
- 4. Initialization and parameters
- 5. Different types of Reservoir Computing
- **6. Practical advices for parameter optimization**
- 7. Summary on ESN
- 8. Application to language processing
- 9. Conceptors: new way to train Reservoirs

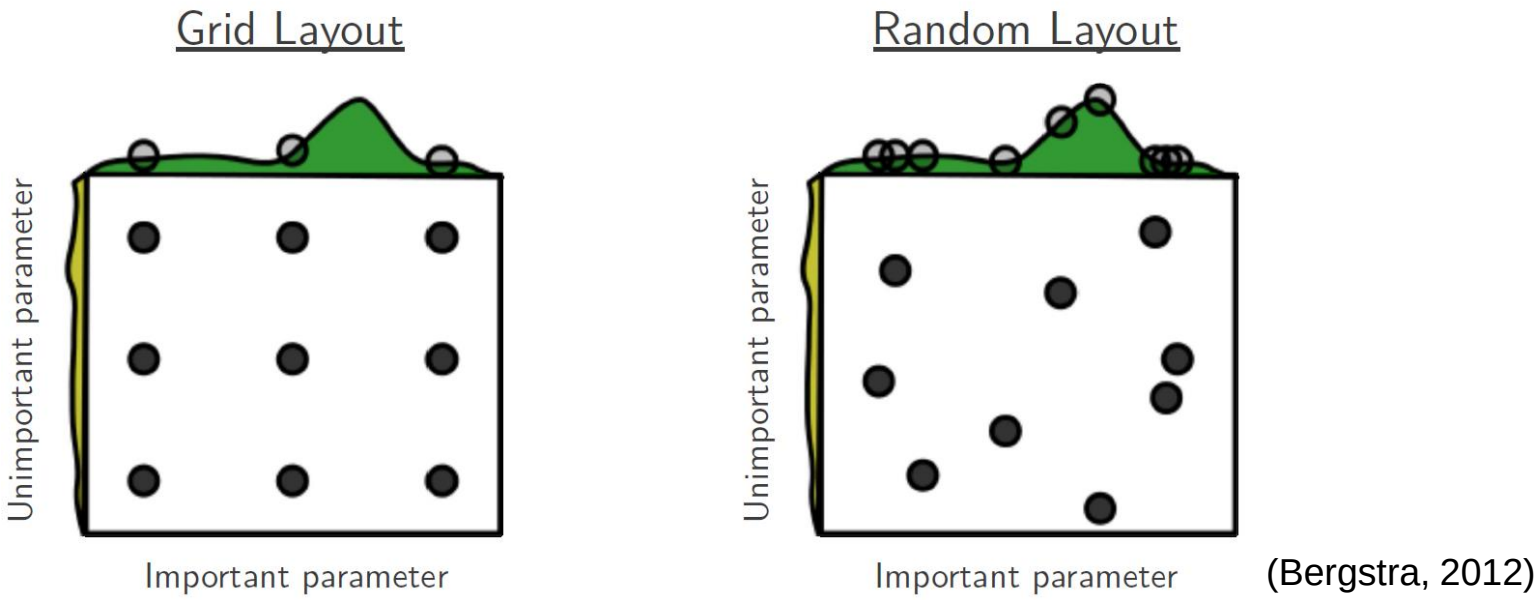
Practical advices for parameter optimization

- For 1, 2, 3 param.
  - Grid search can be used to get an idea of param. landscape
  - but random search also enable to explore this landscape
- ≥ 3 param.
  - Random search or optimization methods
  - e.g. Hyperopt, Evolutionary methods, ...



Practical advices for parameter optimization

- ⇨ grid searches often fails in high dimension spaces
  - (in general) only a few hyperparameters are important to tune
  - « Grid search allocate **too many trials** to the exploration of **dimensions that do not matter**
  - and suffer from **poor coverage in dimensions that are important**. » (Bergstra, 2012)

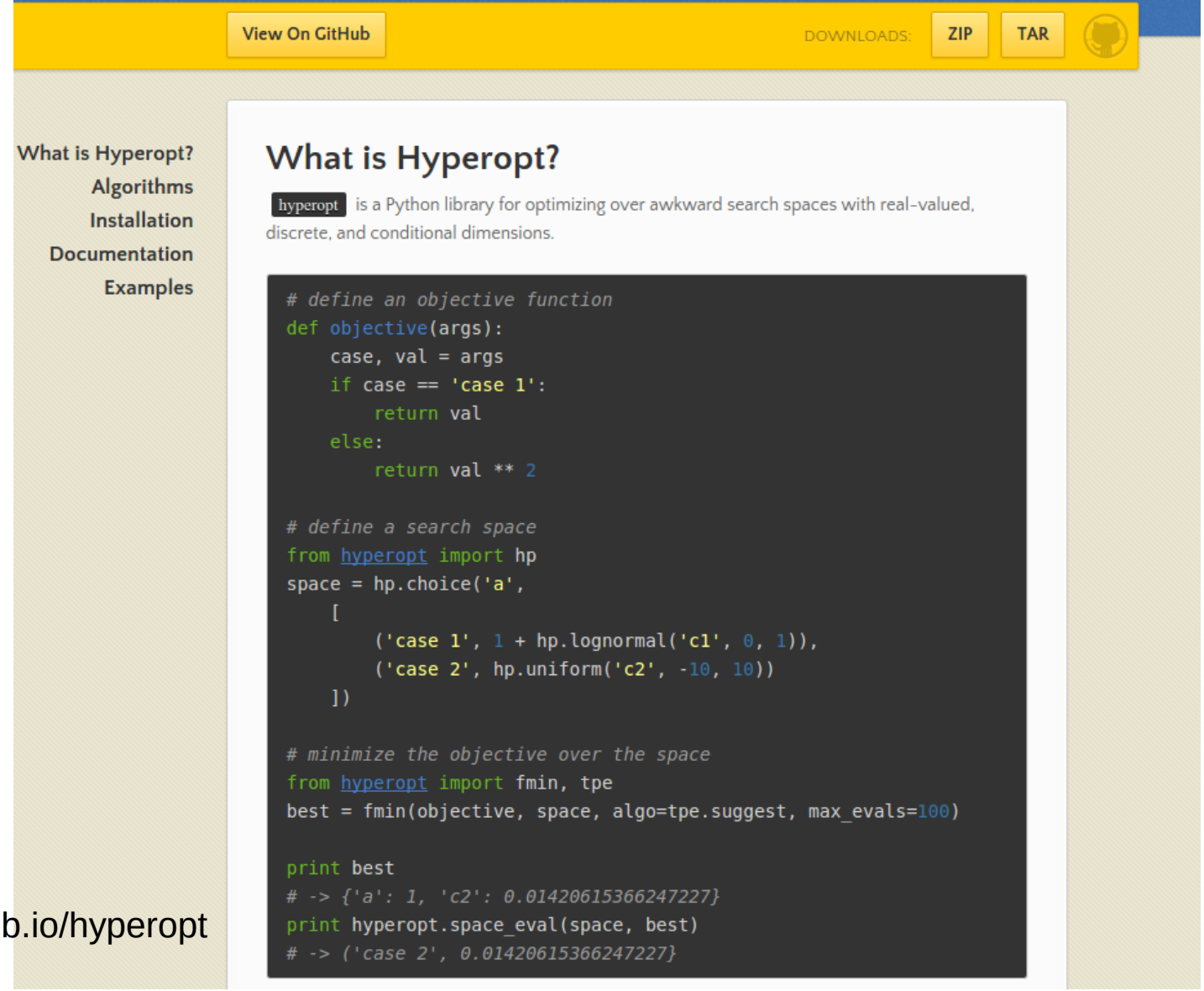


(Bergstra, 2012)

Hyperopt

Distributed Asynchronous Hyperparameter Optimization in Python

View On GitHub DOWNLOADS: ZIP TAR



What is Hyperopt?
Algorithms
Installation
Documentation
Examples

What is Hyperopt?

hyperopt is a Python library for optimizing over awkward search spaces with real-valued, discrete, and conditional dimensions.

```
# define an objective function
def objective(args):
    case, val = args
    if case == 'case 1':
        return val
    else:
        return val ** 2

# define a search space
from hyperopt import hp
space = hp.choice('a',
    [
        ('case 1', 1 + hp.lognormal('c1', 0, 1)),
        ('case 2', hp.uniform('c2', -10, 10))
    ])

# minimize the objective over the space
from hyperopt import fmin, tpe
best = fmin(objective, space, algo=tpe.suggest, max_evals=100)

print best
# -> {'a': 1, 'c2': 0.01420615366247227}
print hyperopt.space_eval(space, best)
# -> ('case 2', 0.01420615366247227}
```

jaberg.github.io/hyperopt

Outline

- 1: Introduction on Echo State Networks (ESN)
- 2. Some examples
- 3. How does it work? Diving in the reservoir
- 4. Initialization and parameters
- 5. Different types of Reservoir Computing
- 6. Practical advices for parameter optimization
- **7. Summary on ESN**
- 8. Application to language processing
- 9. Conceptors: new way to train Reservoirs

Summary

- Fast learning:
  - ***Train only the output layer***
  - Use of a linear method in general
  - Use of sparse matrices (operations can be parallelized)
  - Offline learning done in one pass (not incremental)
- ***Rich temporal non-linear dynamics*** inside the reservoir
  - Many computations are performed in parallel and in a distributed fashion inside the reservoir
  - Output: ***Extract just the information needed***
- Biologically plausible:
  - Learning do not need to "***unfold time***" (as opposed to DL)
  - Online learning methods possible
    - Offline method could be seen as a "shortcut" of online version

Summary

- Some parameters have to be tuned
  - ***Exploration of parameter space*** and expertise necessary to build good reservoir applications (or use Hyperopt toolbox)
  - Main parameters to tune:
    - ***Spectral radius / Input scaling / leak-rate*** / nr. of neurons
    - Ridge parameter (specific to each reservoir instance)
- Stability issues when using feedback
- Difficult to learn long time dependencies
- A ***new paradigm*** for computations inside RNNs
  - All connexions of networks do not need to be trained/tuned
    - → can use a more important number of neurons
  - Sparse random matrices W and W_{in} (5% to 30%)
- Future of reservoir? -> “Conceptors” (Jaeger 2014)

Outline

- 1: Introduction on Echo State Networks (ESN)
- 2. Some examples
- 3. How does it work? Diving in the reservoir
- 4. Initialization and parameters
- 5. Different types of Reservoir Computing
- 6. Practical advices for parameter optimization
- 7. Summary on ESN
- **8. Application to language processing**
- 9. Conceptors: new way to train Reservoirs

Inspiring from Human Language Acquisition

Language acquisition

Grounded Language

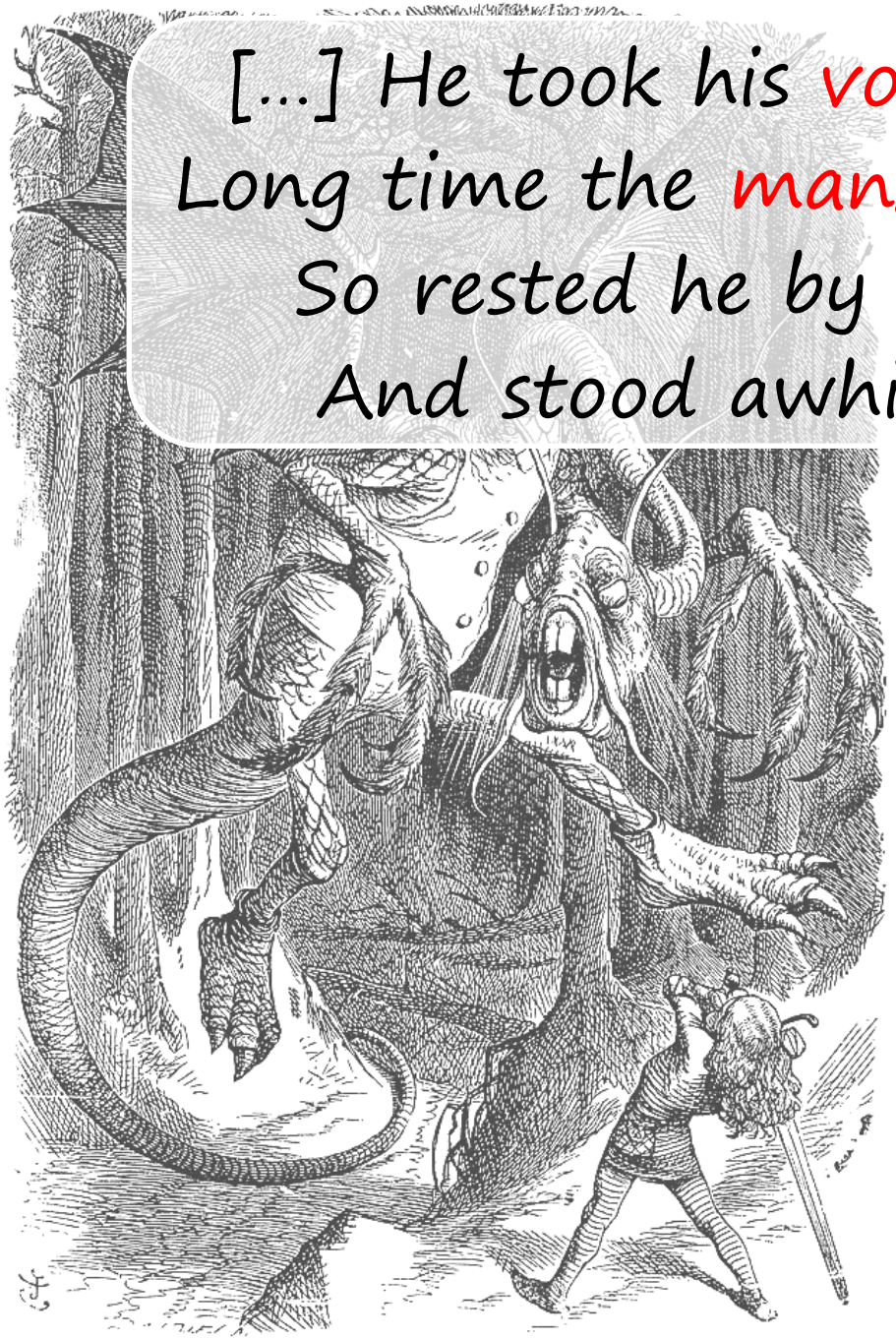
Language Brain Model

Dynamics of acquisition

→ Quickly learn some basic knowledge

[...] He took his vorpal sword in hand:
Long time the manxome foe he sought—
So rested he by the Tumtum tree,
And stood awhile in thought. [...]

[…] He took his vorpal sword in hand:
Long time the manxome foe he sought—
So rested he by the Tumtum tree,
And stood awhile in thought. […]



Jabberwocky, Lewis Carroll
Through the Looking-Glass,
and What Alice Found There (1871)

Task: Thematic Role Assignment

– Representation of meaning (predicate form) :

Predicate (Agent , Object, Recipient)

« The boy gives the ball to the dog »

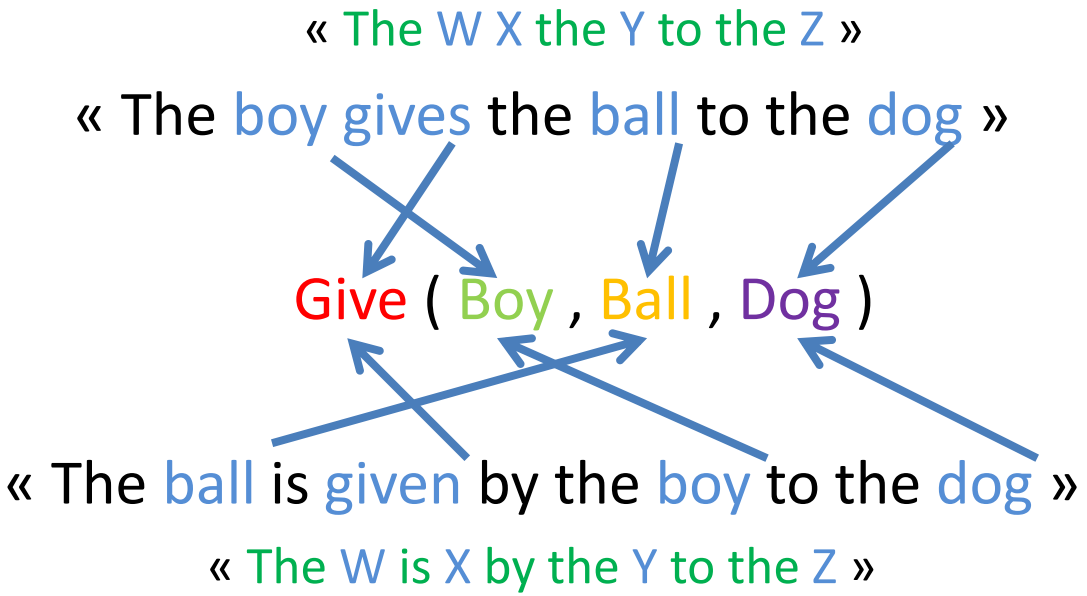
Give (Boy , Ball , Dog)

« The ball is given by the boy to the dog »

Task: Thematic Role Assignment

– Representation of meaning (predicate form) :

Predicate (Agent , Object, Recipient)



Mapping between semantic words and their **roles (** P (A , O, R)**)**

Input: structure of sentence

« The W X -s the Y to the Z »

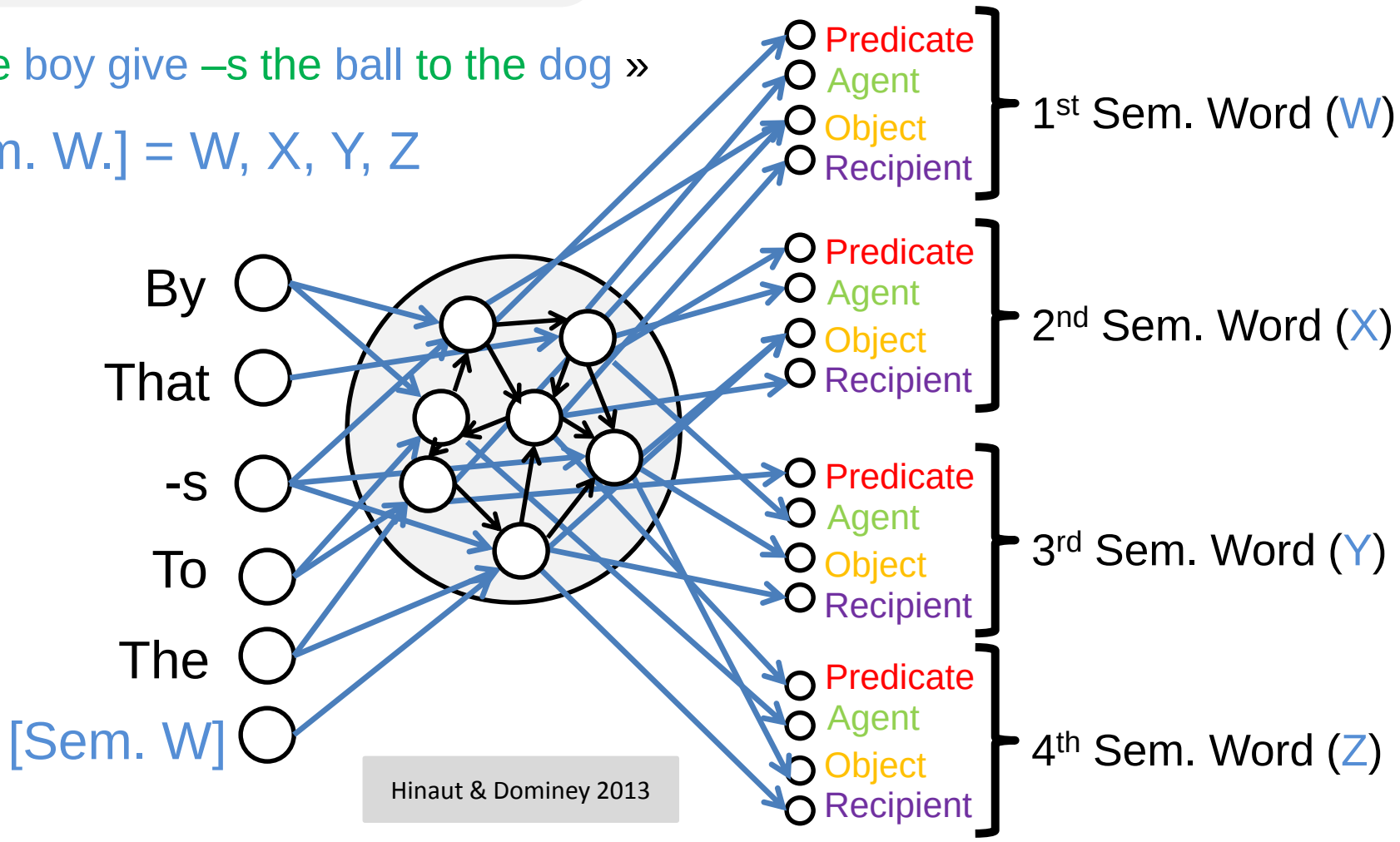
1st 2nd 3rd 4th

Output: meaning (predicate)

Give (Boy , Ball , Dog)

« The boy give –s the ball to the dog »

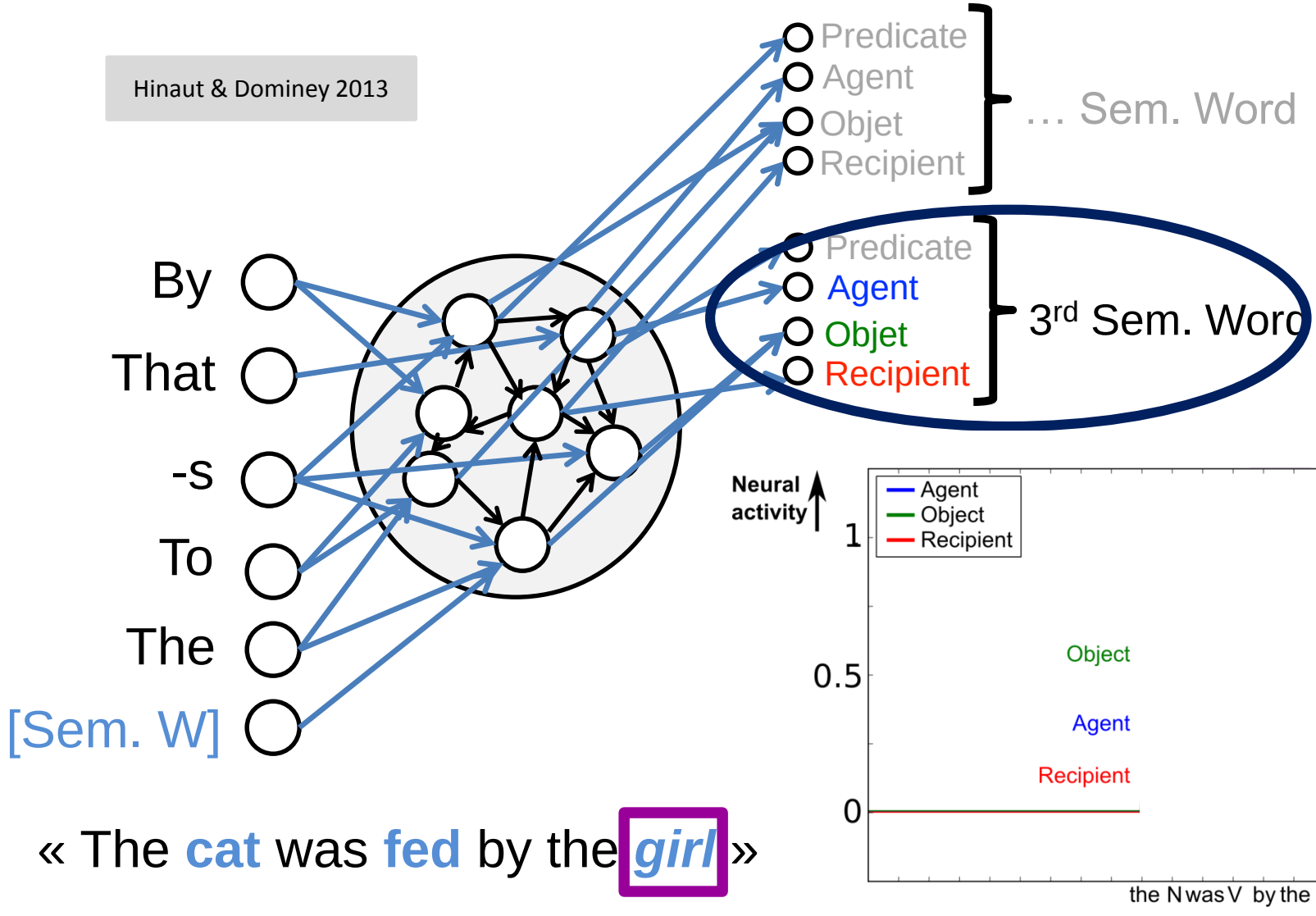
[Sem. W.] = W, X, Y, Z



Hinaut & Dominey 2013

After training … Testing

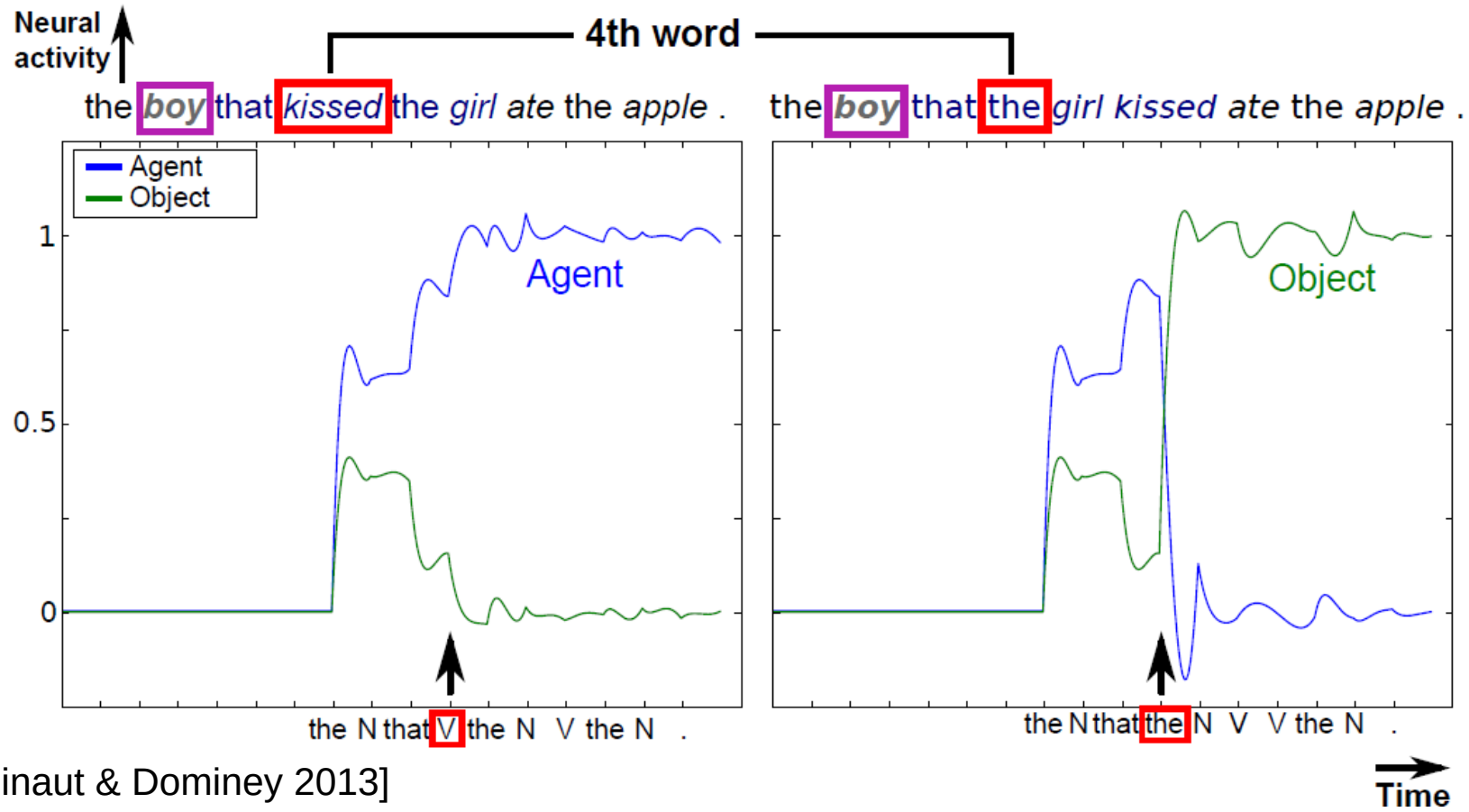
Hinaut & Dominey 2013



« The **cat** was **fed** by the ***girl*** »

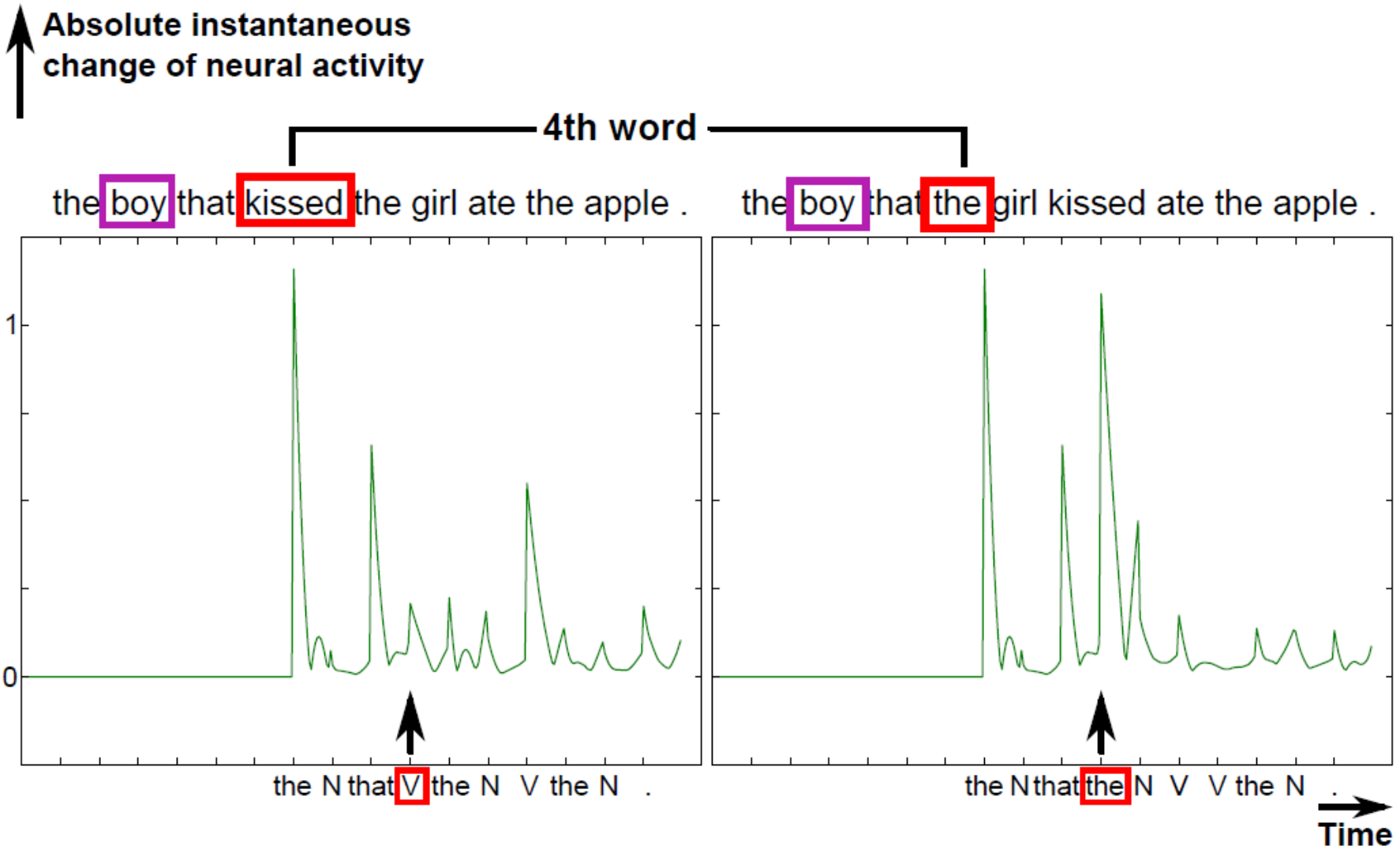
More complex sentences

- On-going roles for the relative clause for the 1st SW: **"boy"**
  - The boy that **kissed** the girl ate the apple. (subject-relative)
  - The boy that **the** girl kissed ate the apple. (object-relative)



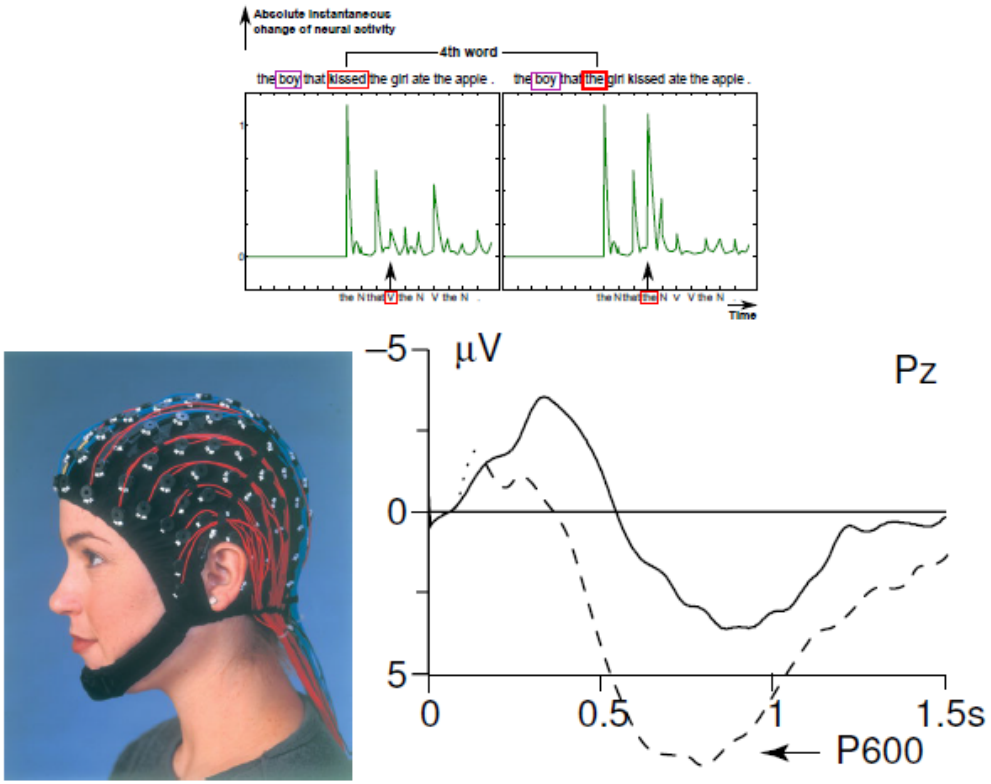
[Hinaut & Dominey 2013]

More complex sentences



[Hinaut & Dominey 2013]

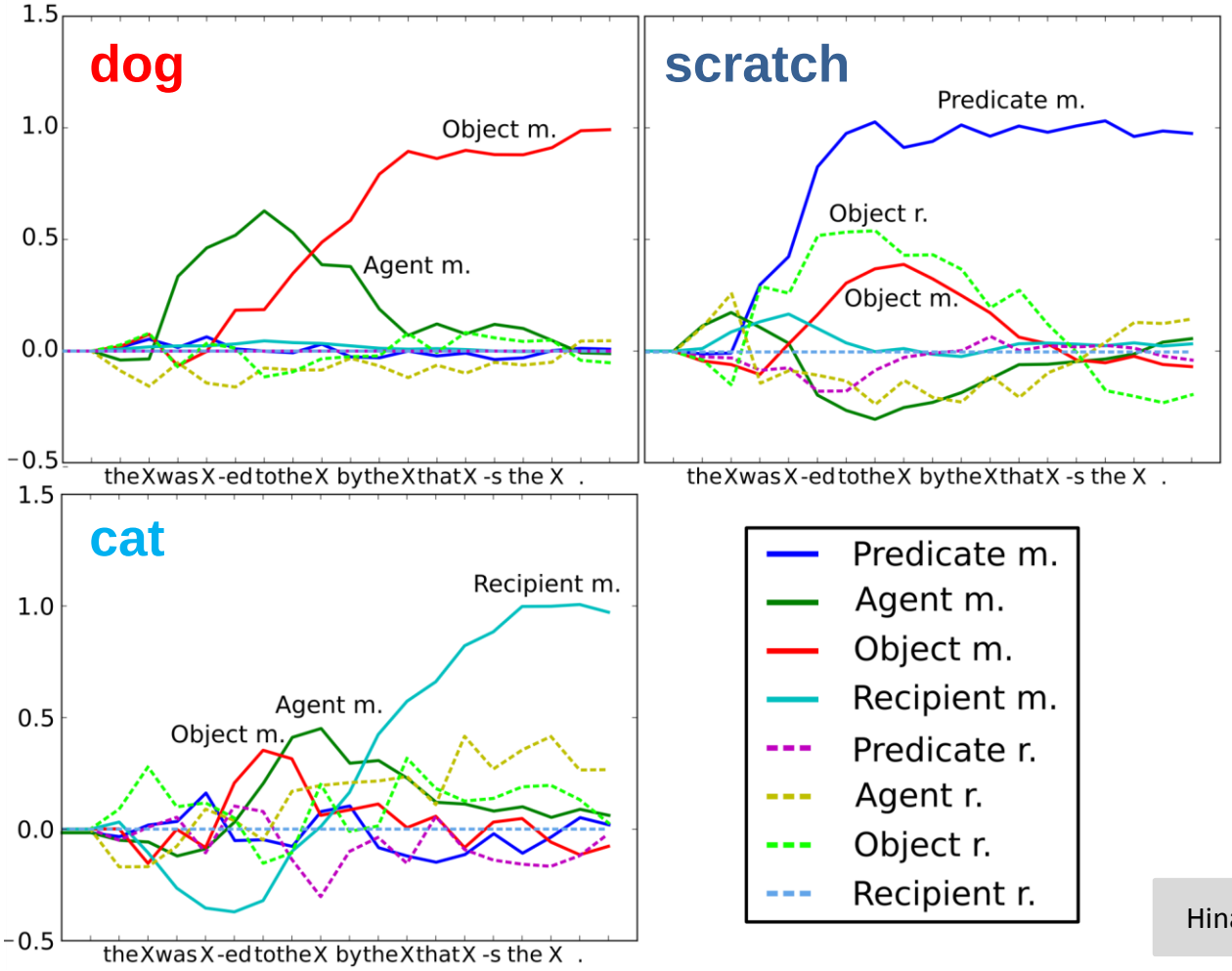
More complex sentences



- In humans:
  - Words indicating low frequency constructions can evoke a P600
- In the model:
  - Similarly, a significant change in output activity occurs at the onset of the word indicating unfrequent constructions

With online training proc.

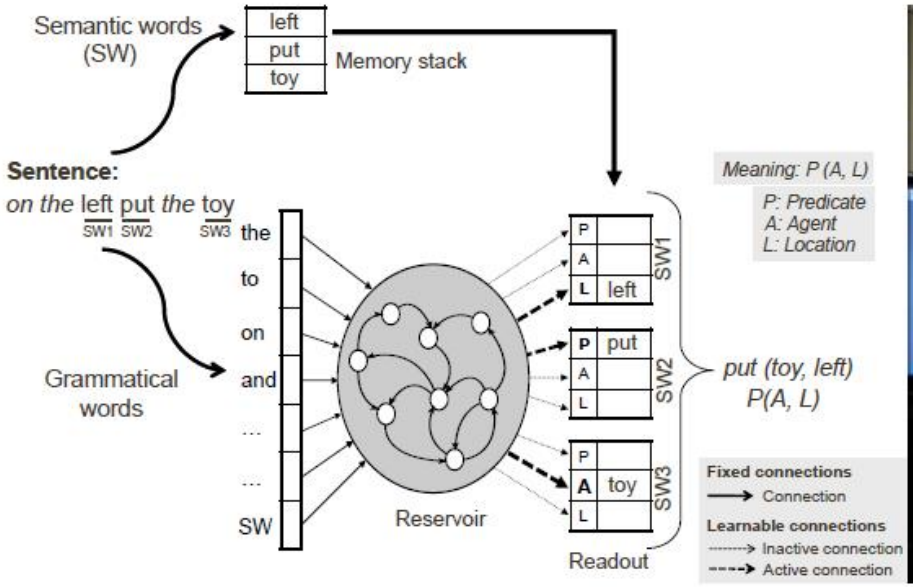
RobotDoC
Robotics for Development of Cognition

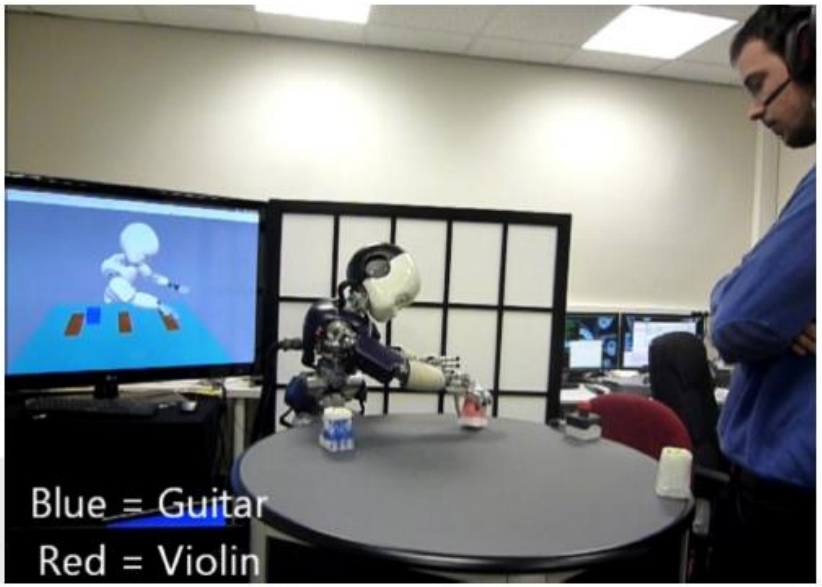


Hinaut & Wermter 2014

The ***dog*** was ***scratch*** –ed by the ***cat*** that ***see*** –s the ***mouse*** .

Robotic Implementations





[Hinaut et al. 2014, Hinaut et al. 2015]

- Video links:
  - https://youtu.be/AUbJAupkU4M
  - https://youtu.be/3ZePCuvygi0
  - https://www.youtube.com/watch?v=FpYDco3ZgkU

Some sentences said by users

- (5) point the triangle before grasping the circle
- (20) put the cross to the left before grasping the circle
- (92) point to the cross twice
- (198) before you grasp the cross please grasp the triangle
- (214) before pushing the triangle to the middle please push the cross to the right
- (230) grasp the circle and then point to it
- (245) touch the triangle then move it to the left
- (260) the cross touch it
- (268) point to the circle after having grasped it

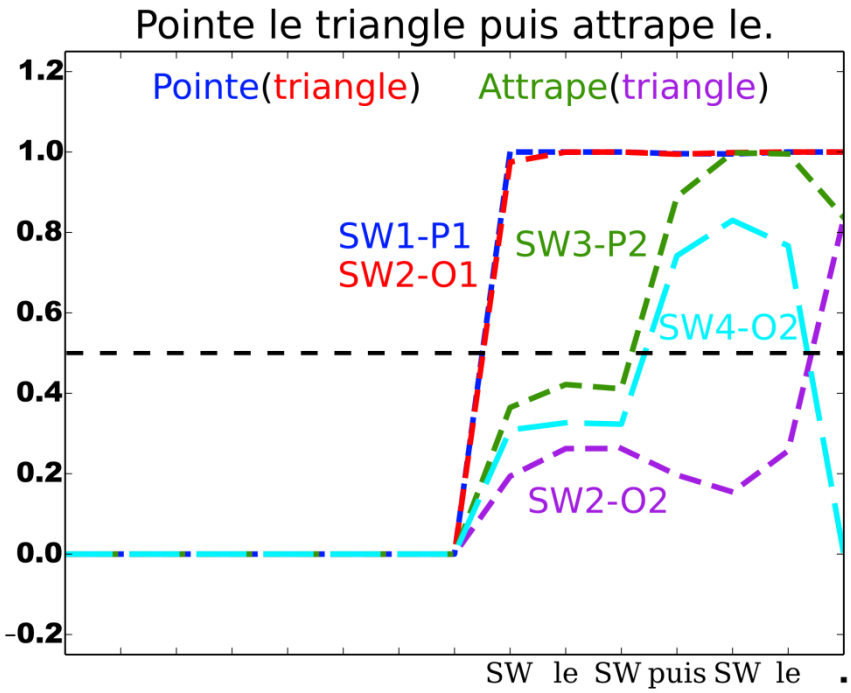
[Hinaut et al. 2014]

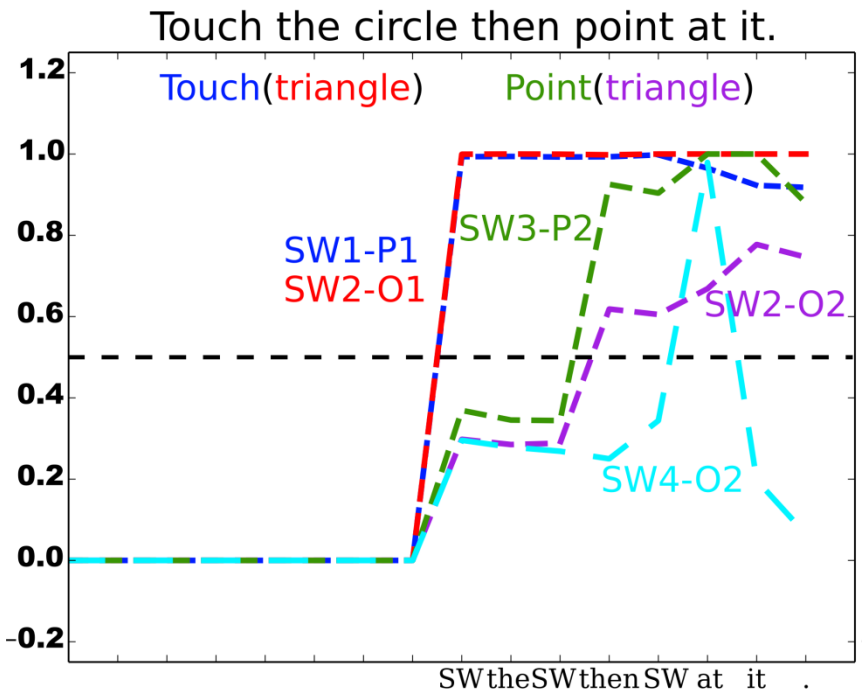
Training data set example

1st command; ***2nd command***	Do this then *do that*
Point(triangle); ***Grasp(circle)***	point the triangle before *grasping the circle*
Grasp(triangle); ***Grasp(cross)***	before *you grasp the cross* please grasp the triangle
Point(cross); ***Point(cross)***	*point to the cross* twice
Touch(triangle); ***Move(triangle, left)***	touch the triangle then *move it to the left*

[Hinaut et al. 2014]

Learning French & English with same Res.





[Hinaut et al. 2015]

Proof-of-concept for 15 languages

(with different word order)

1) answer phone ; answer the phone
2) geh an Telefon ; geh an das Telefon
3) contesta teléfono ; contesta el teléfono
4) réponds téléphone ; réponds au téléphone
5) rispondi telefono ; rispondi al telefono
6) contesta telèfon ; contesta el telèfon
7) 接电话;接那-通电话
8) 接電話;接那-通電話
9) jawab telefon ; jawab -kan panggilan telefon itu
10) cevap ver telefon ; telefon -a cevap ver
11) uchal phone ; phone uchal
12) dho javaab phon ; phon ka javaab dho
13) telefono erantzun ; telefono -a erantzun
14) javab telfon ; telfon ra javab bede
15) atenda telefone ; atenda o telefone
16) вдигни телефон ; вдигни телефон -а

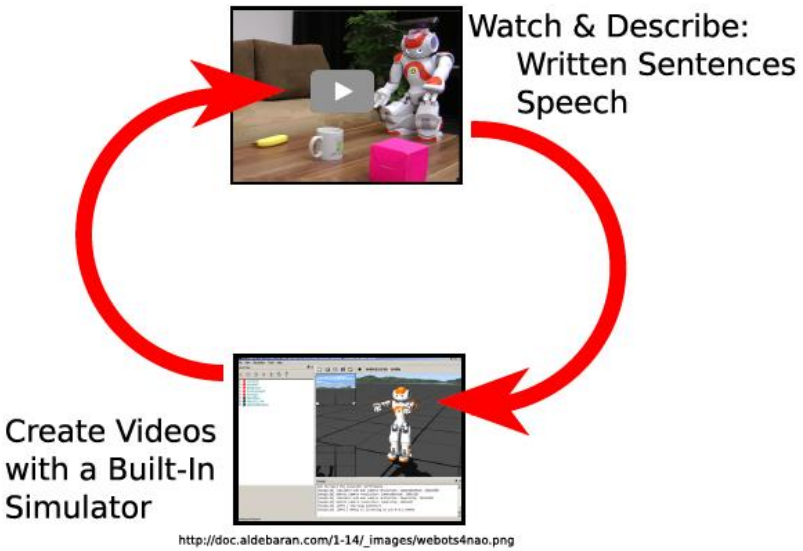
1) English
2) German
3) Spanish
4) French
5) Italian
6) Catalan
7) Simpl. Mandarin
8) Trad. Mandarin
9) Malay
10) Turkish
11) Marathi
12) Hindi
13) Basque
14) Persian
15) Portuguese
16) Bulgarian

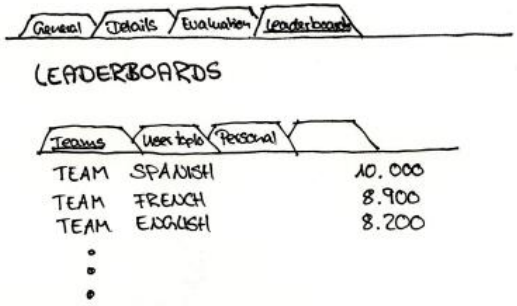
[Hinaut et al. 2018]

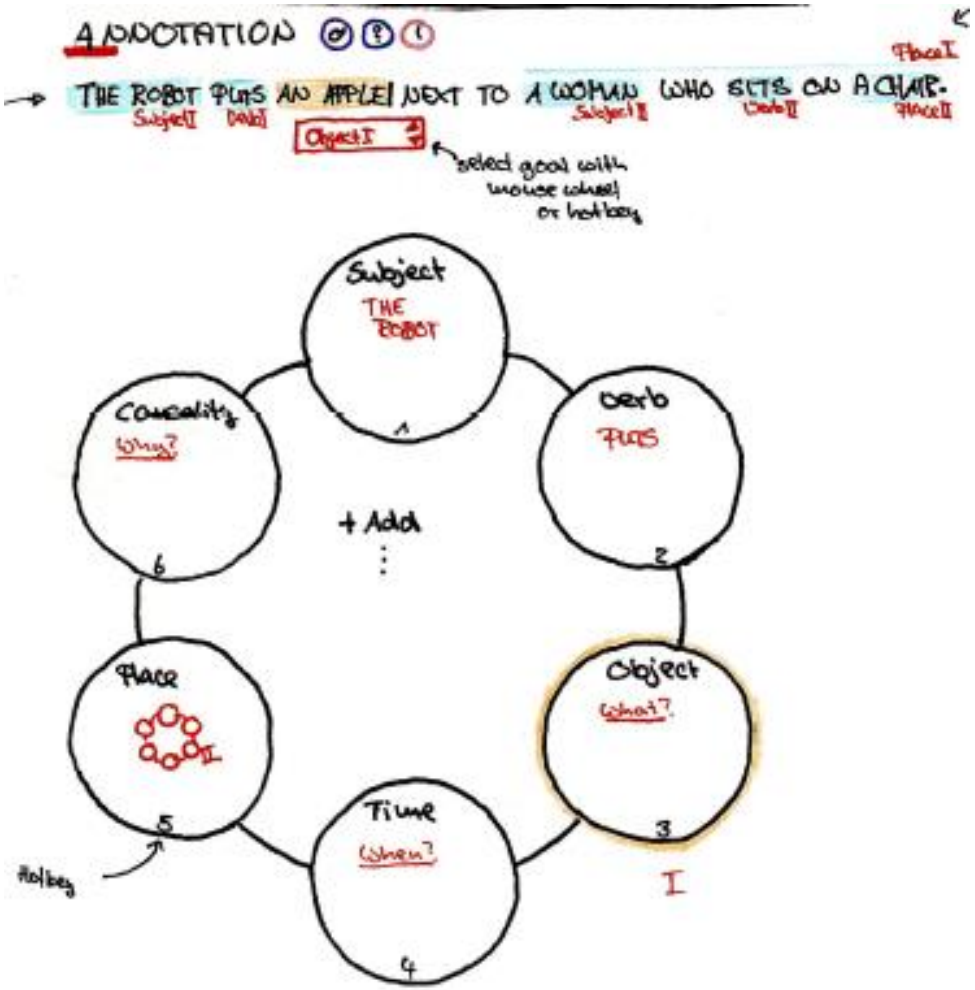
Summary on Language Model

- Generic neural circuit for sequence processing
  - Linear classifiers decode high-dimensional representations
- Online and anytime (can be probed before the end of the sentence)
- Generalize (to unseen constructions ranging in size from 20 to 90,000)
  - Also with sent. with unkown words
- Enable robots to learn non-stereotypical sentences
- The same instance of the circuit can learn from different inputs (French & English)
- Proof of concept for 15 Asian-European languages
- ROS module available

Crowdsourcing website to collect data

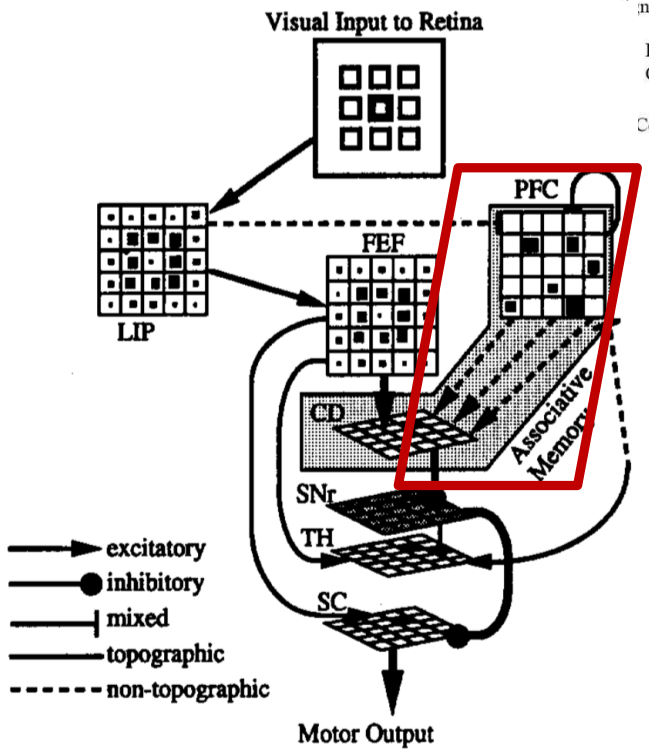




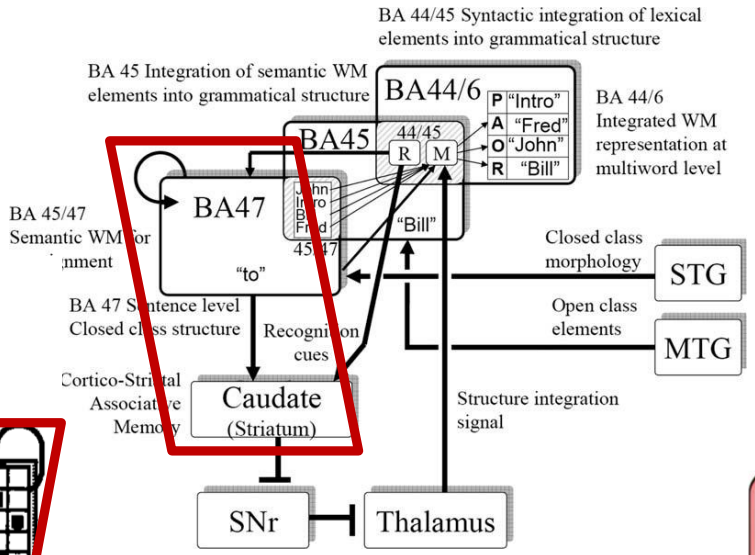


- To participate or help
  - xavier.hinaut@inria.fr
  - twiefel@informatik.uni-hamburg.de

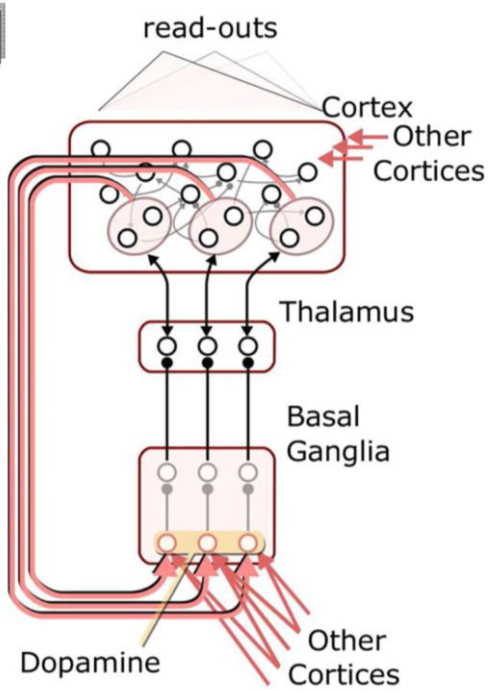
Other computational neuroscience models using reservoirs



Dominey 1995



Dominey 2009

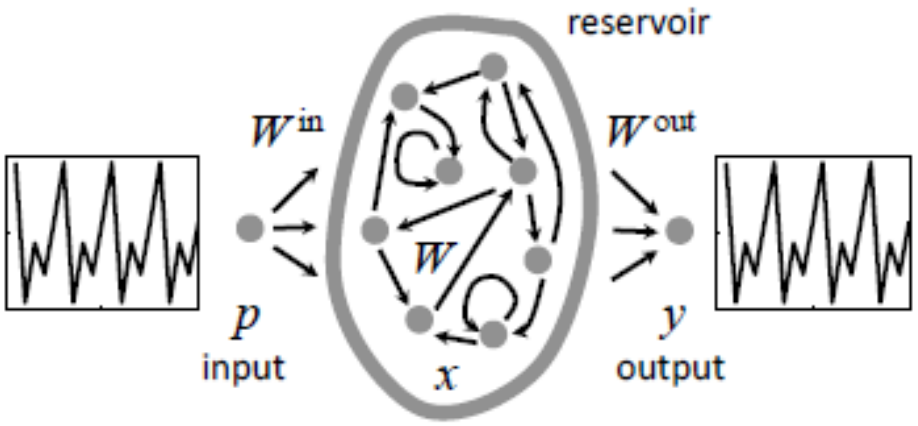


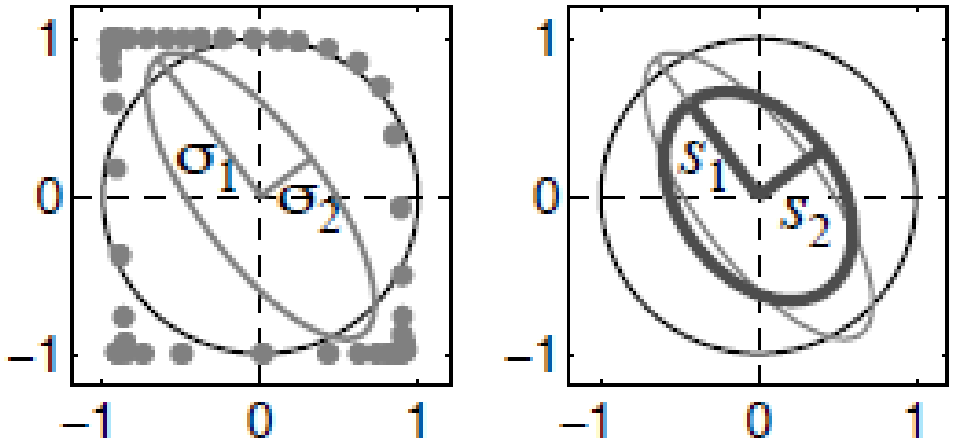
Mannella & Baldassarre 2015

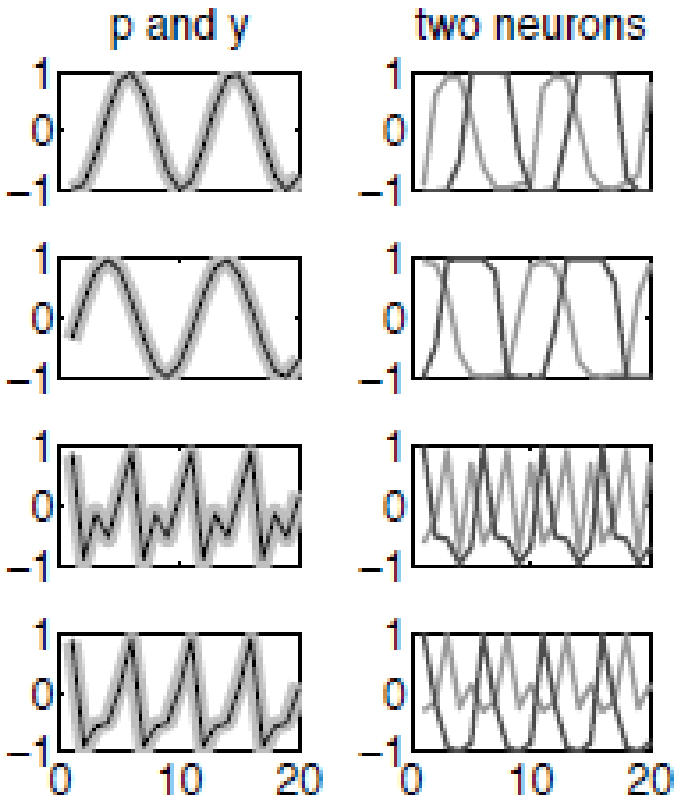
Outline

- 1: Introduction on Echo State Networks (ESN)
- 2. Some examples
- 3. How does it work? Diving in the reservoir
- 4. Initialization and parameters
- 5. Different types of Reservoir Computing
- 6. Practical advices for parameter optimization
- 7. Summary on ESN
- 8. Application to language processing
- **9. Conceptors: new way to train Reservoirs**

Conceptors: new way of training Reservoirs

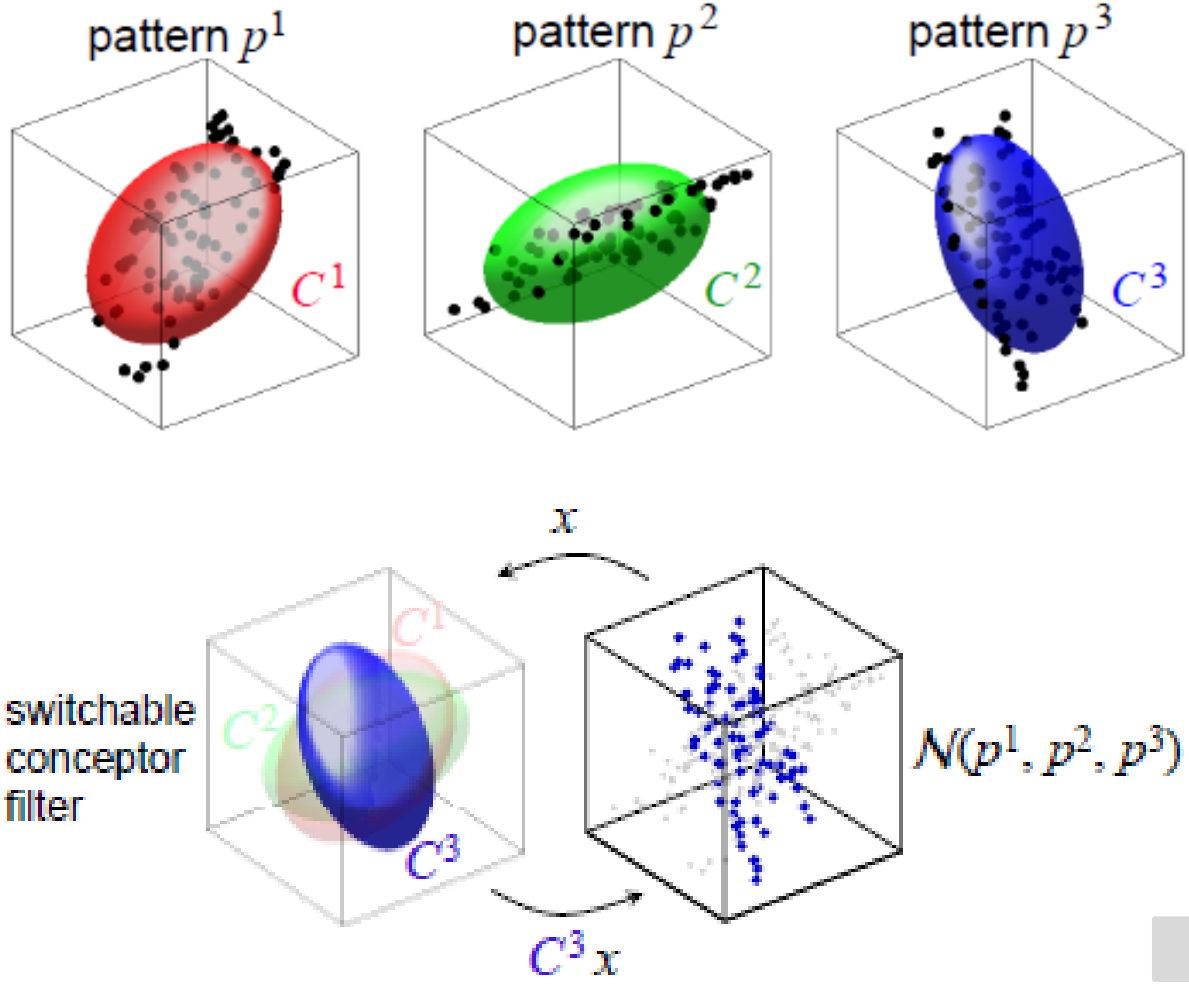






Jaeger 2014

A conceptor C^j imposes dynamics on the RNN



Jaeger 2014

How to build conceptors? (1) build W

W is computed as follows. In two separate runs, the two patterns are fed into the initial reservoir via

$$\mathbf{x}^j(n+1) = \tanh(W^* \mathbf{x}^j(n) + W^{\text{in}} p^j(n) + \mathbf{b}), \quad j = 1, 2;\ n = 0, \ldots, L. \tag{3}$$

Figure 1 (middle panels) shows the activation traces of the three neurons when the reservoir is driven with either pattern.

Then W is computed to minimize the quadratic loss

$$\sum_{j=1,2} \sum_{n=n_0+1,\ldots,L} \|W^* \mathbf{x}^j(n) + W^{\text{in}} p^j(n) - W \mathbf{x}^j(n)\|^2, \tag{4}$$

where only network states for times after n_0 are used (in order to allow for washing out the arbitrary intial state $\mathbf{x}(0)$ according to the echo state property). This again amounts to a linear regression. If this is achieved with a small training error, one will obtain similar network updates from states $\mathbf{x}^1(n)$ or $\mathbf{x}^2(n)$ with either the original input-driven update rule, or with an input-free update rule that employs W instead of W^*:

$$\tanh(W^* \mathbf{x}^j(n) + W^{\text{in}} p^j(n) + \mathbf{b}) \approx \tanh(W \mathbf{x}^j(n) + \mathbf{b}). \tag{5}$$

I call this procedure to transform the initial random weights W^* to W *loading* the patterns into the reservoir.

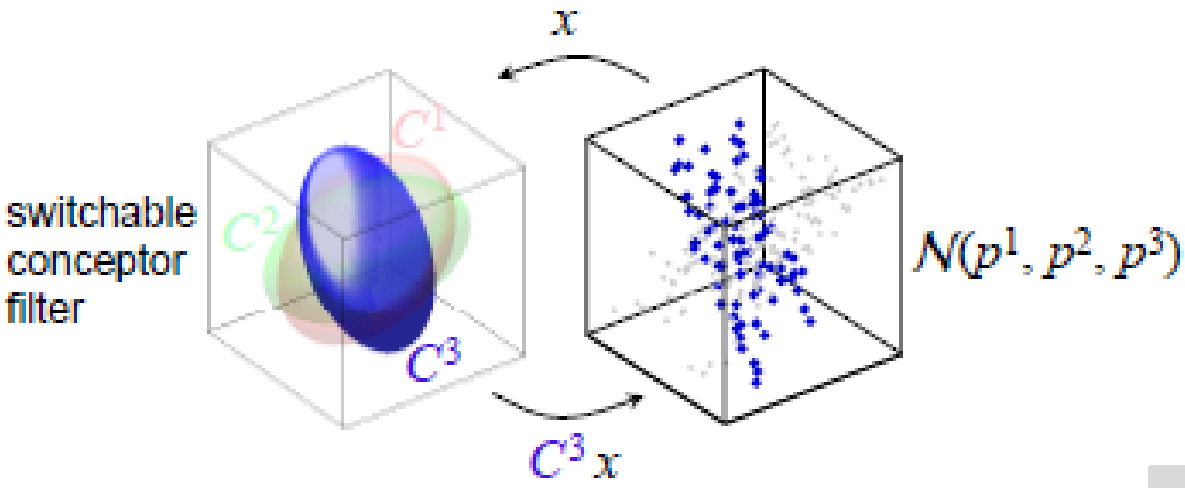
Jaeger 2017

(2) insert the conceptor Cj in the state up. eq.

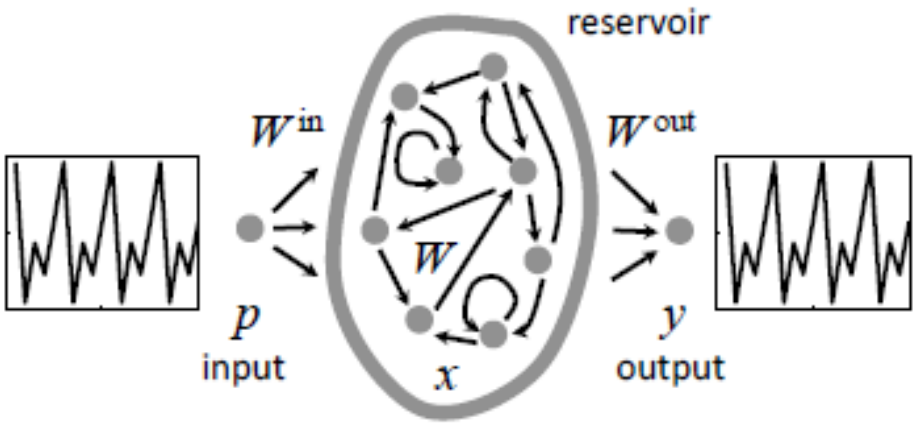
Here conceptors enter the stage. Each loaded pattern p^j is associated with an $N \times N$ sized *conceptor* matrix C^j which at recall time is inserted into the state update loop via

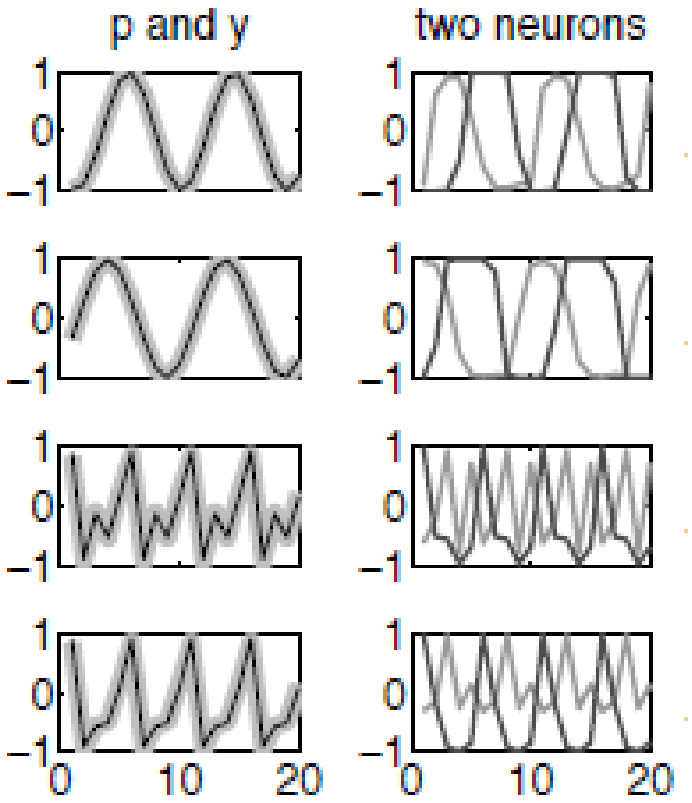
$$\mathbf{x}(n+1) = C^j \tanh(W\,\mathbf{x}(n) + \mathbf{b}).$$

The matrix C^j acts as a filter that leaves states $\mathbf{x}^j(n)$ from the state pattern associated with pattern p^j essentially unchanged, but suppresses state components of states $\mathbf{x}^{j'}(n)$ associated with other patterns $p^{j'}$. Stated differently, C^j should act like the identity matrix for states $\mathbf{x}^j(n)$, but like the null matrix for state components that are not typical for states $\mathbf{x}^j(n)$.



(3) train the readout to obtain back the input pattern





By loading the conceptor C^j we will obtain back the input pattern p^j
(or a denoised version if it was noisy)

Conceptors: morphing between conceptors

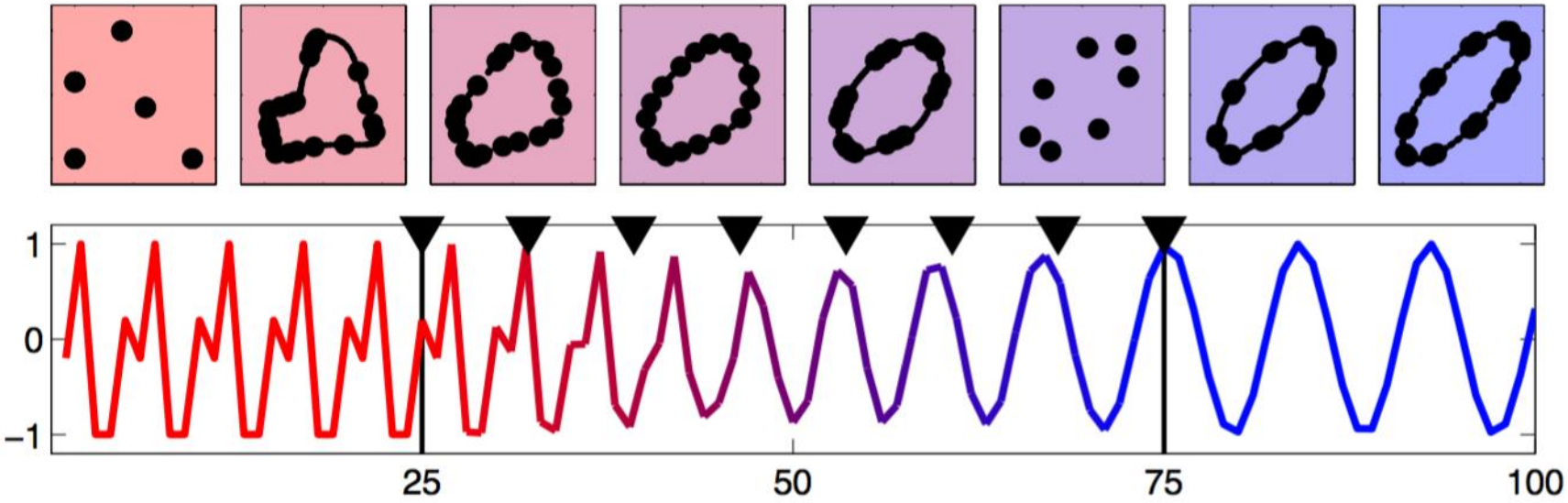


Figure 3: Morphing from an integer-5-periodic pattern p^1 to an irrational-period sine p^2 from time $n = 25$ to $n = 75$. Top: delay-embedding plots of signals $y(n)$ obtained with fixed conceptor mixes (indicated by triangle markers in center panel). Bottom: Output signal $y(n)$.

Conceptors: morphing between conceptors

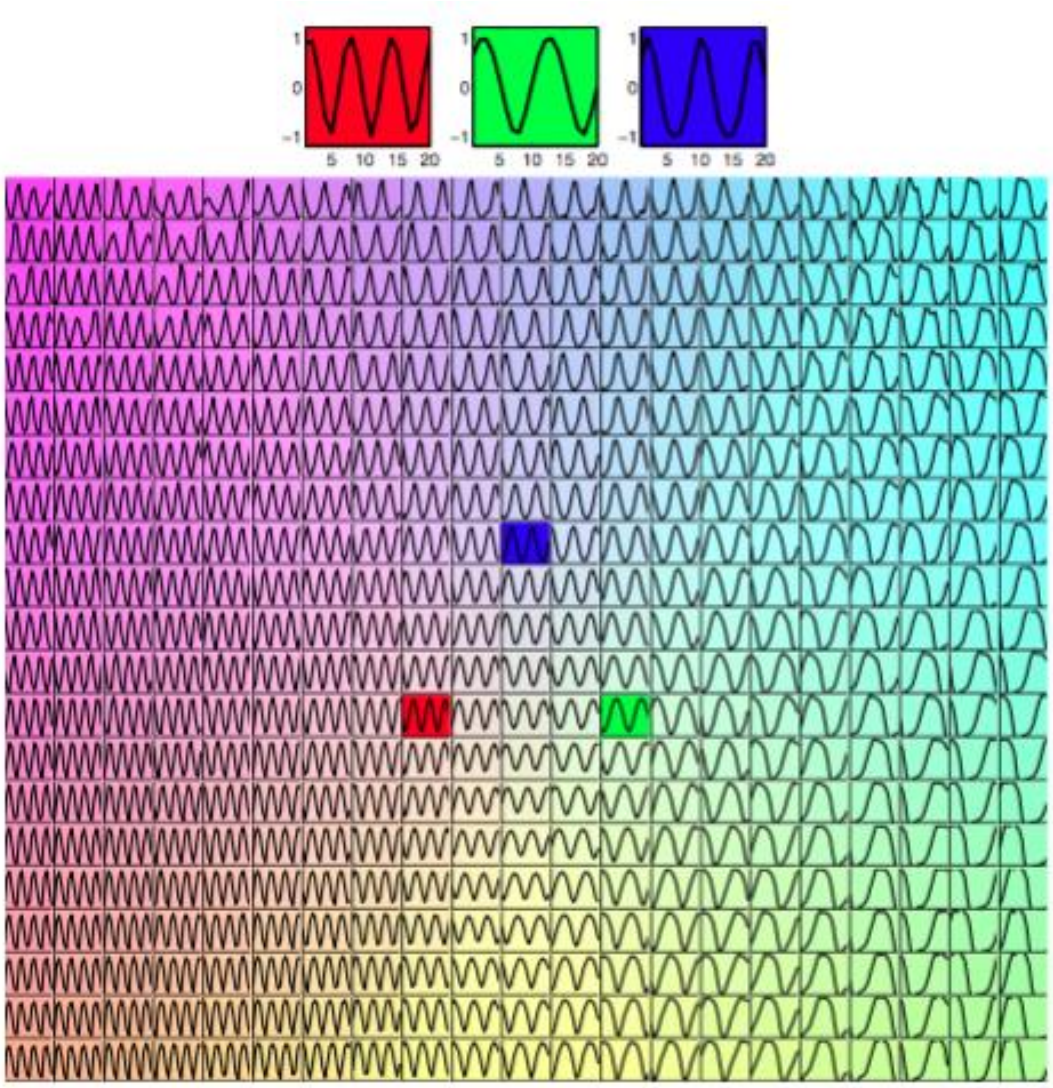


Figure 4: Morphing between and beyond three patterns from a 2-parametric family.

Jaeger 2017

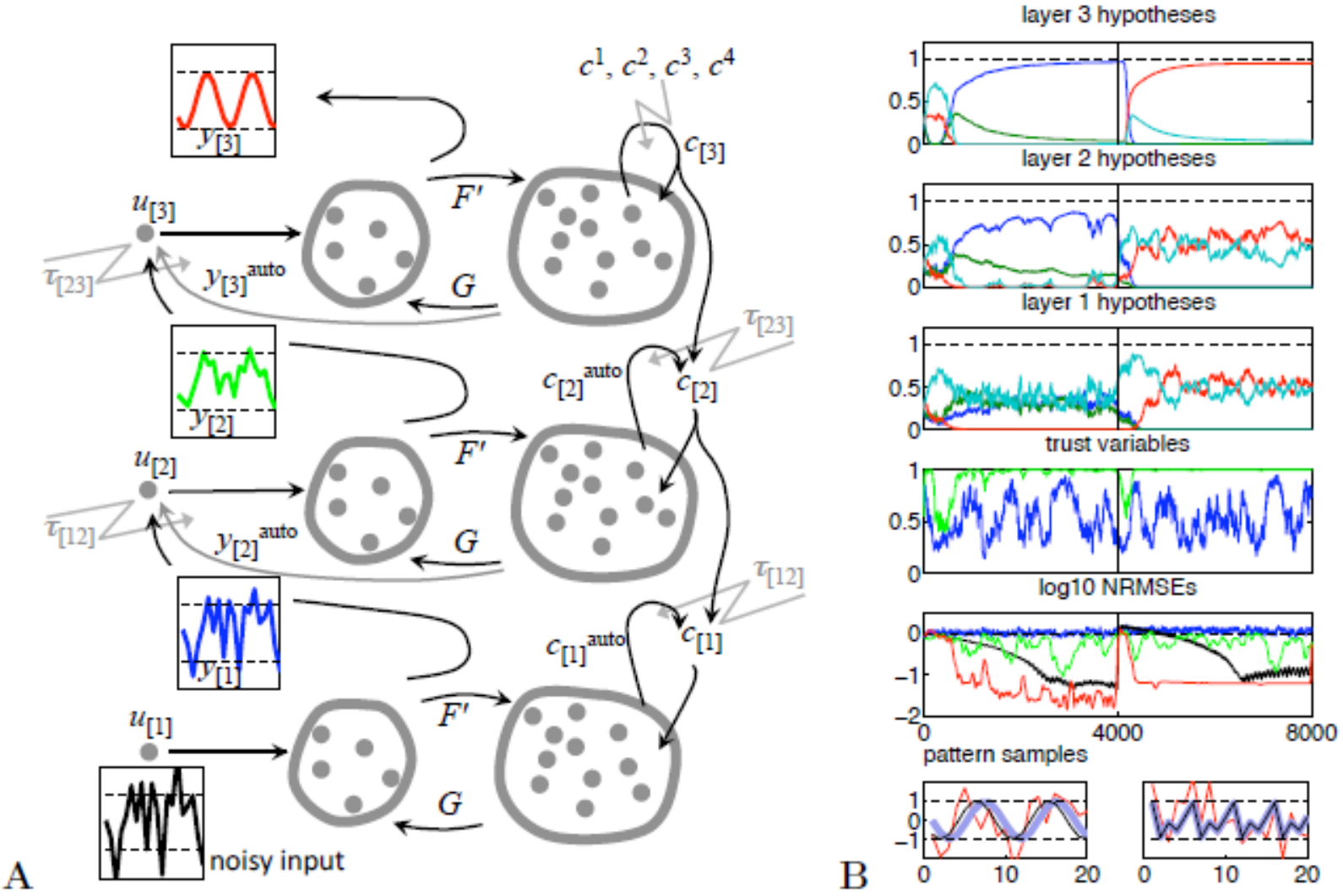
Conceptors: morphing between conceptors

Figure 6: Snapshots from a human motion re-generation video created from a conceptor-controlled RNN. First row: standing up from a stool and starting a slow walk, second row: kneeling down and starting to crawl, third row: cartwheel.

- https://www.youtube.com/watch?v=DkS_Yw1ldD4

Jaeger 2017

Conceptors: bio-inspired implementation of a hierarchical denoiser



Jaeger 2014

Summary on ESN / Reservoir Computing

- Quick way to train RNN without backpropagation
- Used in various applications in industry
  - e.g. on system with low ressources
  - good for robots to learn online
  - … (make a search on google scholar)
- A lot of variant
  - unsupervised learning inside reservoir
  - other design methods for rec. weights
  - … (make a search on google scholar)
- Research on “physical-ware” reservoir
  - e.g. photonic reservoirs
- Good models for computational neuroscience

Try it!

- A simple and flexible code for ESN:
  - https://github.com/neuronalX/Funky_Reservoir
- I will add more features in the next months/years
  - e.g. with conceptors
- Contact: xavier.hinaut@inria.fr

Further reading on ESN

- (Introduction) Jaeger H (2007) Echo state network. Scholarpedia 2: 2330. http://www.scholarpedia.org/article/Echo_state_network
- (*Tutorial: comparison with other RNN methods*) H. Jaeger (2002): Tutorial on training recurrent neural networks, covering BPPT, RTRL, EKF and the "echo state network" approach. GMD Report 159, German National Research Center for Information Technology, 2002 (48 pp.) (Fifth revision: Dec 2013) http://minds.jacobs-university.de/sites/default/files/uploads/papers/ESNTutorialRev.pdf
- (*Practical guide*) Lukoševičius M (2012) A Practical Guide to Applying Echo State Networks. In: Montavon G, Orr GB, Müller K-R, editors. Neural Networks: Tricks of the Trade, Reloaded. Springer. http://organic.elis.ugent.be/publications/practical_echo_state_networks

References (Optional)

- (Specific to l*eaky reservoirs*) Jaeger H, Lukoševičius M, Popovici D, Siewert U (2007) Optimization and applications of echo state networks with leaky-integrator neurons. Neural networks 20: 335–352.
- Conceptors
  - Jaeger H. (2014) Conceptors: an easy introduction. arXiv:1406.2671v1
  - Jaeger, Herbert. "Controlling recurrent neural networks by conceptors." *arXiv preprint arXiv:1403.3369* (2014).
  - Jaeger, Herbert. "Using conceptors to manage neural long-term memories for temporal patterns." *Journal of Machine Learning Research* 18.13 (2017): 1-43.
  - Video: https://www.youtube.com/watch?v=DkS_Yw1ldD4

Further reading on ESN

- (Applied to robot language acquisition) [Hinaut et al. 2015] Hinaut X, Twiefel J, Petit M, Dominey PF, Wermter S (2015) A Recurrent Neural Network for Multiple Language Acquisition: Starting with English and French. Proceedings of the NIPS Workshop on Cognitive Computation: Integrating Neural and Symbolic Approaches. Montreal, Canada, December 11-12, 2015. http://ceur-ws.org/Vol-1583/CoCoNIPS_2015_paper_14.pdf

- [Explanatory Video of HRI system] Humanoidly Speaking – Learning about the world and language with a humanoid friendly robot. IJCAI 2015 Video competition. www.youtube.com/watch?v=FpYDco3ZgkU

Useful toolboxes (Optional)

- (Python code to give a try) [Lukoševičius, minimalESN.py] http://minds.jacobs-university.de/mantas/code
- (Parameter search toolbox) Bergstra J, Yamins D, Cox DD. (2013) Hyperopt: A Python Library for Optimizing the Hyperparameters of Machine Learning Algorithms. Proc. SCIPY 2013.

References (Optional)

- (*Review*) Lukoševičius M, Jaeger H (2009) Reservoir computing approaches to recurrent neural network training. Computer Science Review 3: 127–149. doi:10.1016/j.cosrev.2009.03.005 http://organic.elis.ugent.be/publications/2009-reservoir-computing-approaches-recurrent-neural-network-training

- (*Application to robotic control*) Reinhart RF (2011) Reservoir computing with output feedback Universität Bielefeld. http://d-nb.info/1019275367/34

- (*Intuitive guide for understanding reservoirs*) Verstraeten D (2009) Reservoir Computing: computation with dynamical systems. PhD Thesis, Electronics and Information Systems, University of Ghent. http://organic.elis.ugent.be/publications/submitted-reservoir-computing-computation-dynamical-systems

References (Optional)

- (*Online reservoir with feedback*) Sussillo D, Abbott LF (2009) Generating coherent patterns of activity from chaotic neural networks. Neuron 63: 544–557.
- (*Online reservoir with reinforcement-like learning*) Hoerzer GM, Legenstein R, Maass W (2012) Emergence of Complex Computational Structures From Chaotic Neural Networks Through Reward-Modulated Hebbian Learning. Cerebral Cortex.
- (*Theorical approach*) Legenstein R, Maass W (2007) Edge of Chaos and Prediction of Computational Performance for Neural Circuit Models. Neural Networks 20: 323–334.

References (Optional)

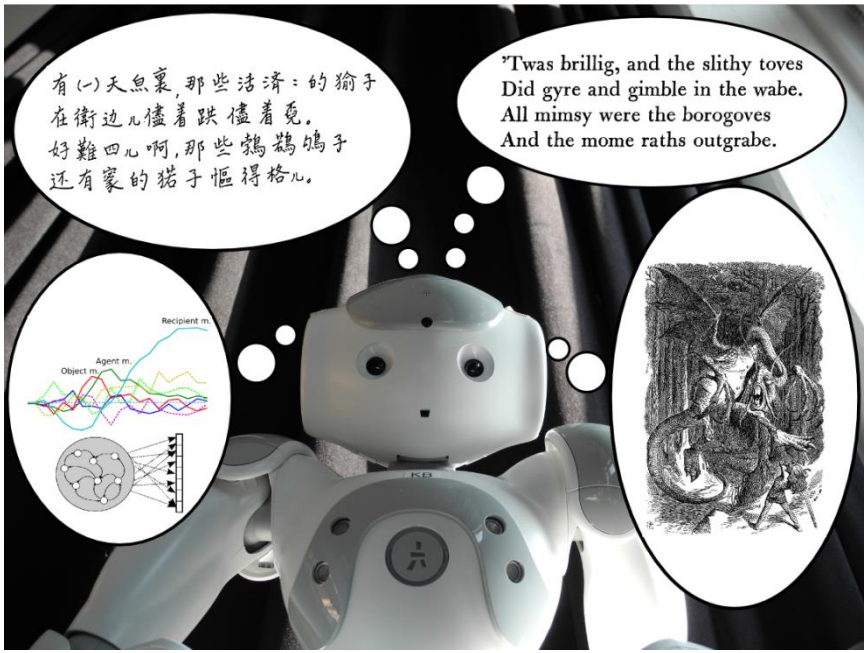
- [Jaeger & Hass 2004] Jaeger H, Haas H (2004) Harnessing nonlinearity: predicting chaotic systems and saving energy in wireless communication. Science 304: 78–80.
- [Maass 2002] Maass, W., Natschläger, T., & Markram, H. (2002). Real-time computing without stable states: A new framework for neural computation based on perturbations. Neural computation, 14(11), 2531-2560.
- [Dominey 1995] Dominey, P. F. (1995). Complex sensory-motor sequence learning based on recurrent state representation and reinforcement learning. Biological cybernetics, 73(3), 265-274.
- [Buonomano & Merzenich 1995] Buonomano D V, Merzenich M (1995) Temporal information transformed into a spatial code by a neural network with realistic properties. Science 267: 1028–1030.

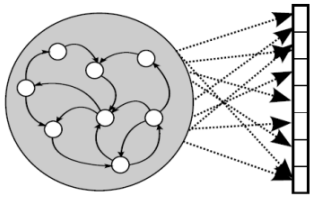
References (Optional)

- Applied to language:
- Hinaut X, Dominey PF (2013) Real-Time Parallel Processing of Grammatical Structure in the Fronto-Striatal System: A Recurrent Network Simulation Study Using Reservoir Computing. PLoS ONE 8(2): e52946. http://www.plosone.org/article/info%3Adoi%2F10.1371%2Fjournal.pone.0052946
- Hinaut X, Petit M, Pointeau G and Dominey PF (2014) Exploring the acquisition and production of grammatical constructions through human-robot interaction with echo state networks. Front. Neurorobot. 8:16. http://journal.frontiersin.org/article/10.3389/fnbot.2014.00016/full
- Hinaut X., Wermter S. (2014) An Incremental Approach to Language Acquisition: Thematic Role Assignment with Echo State Networks. In Artificial Neural Networks and Machine Learning–ICANN 2014. http://www.informatik.uni-hamburg.de/wtm/ps/Hinaut_ICANN2014_CR.pdf
- Hinaut X (2013) Recurrent Neural Networks for Abstract Sequence and Grammatical Structure Processing, with an Application to Human-Robot Interaction. PhD Thesis, University of Lyon, France.

Sequence processing in brains, machines & robots

RobotDoC
Robotics for Development of Cognition





Campus Paris Saclay
Fondation de Cooperation Scientifique