Chaotic Neural Networks Documentation

Release 0.0.1

Younesse Kaddar

Jun 14, 2018

Contents:

CHAPTER 1

Package `chaotic_neural_networks`

1.1 Implementation of D. Sussillo and L.F. Abbott's 2009 article

- iPython notebook
- Github
- Report

1.2 Utilities – Target functions

`chaotic_neural_networks.utils.`**`PCA`**(*data*, *nb_eig=8*, *return_matrix=True*, *return_eigenvalues=True*)

Principal Component Analysis (PCA) to compute the `nb_eig` leading principal components.

Parameters

- **`data`** (*`(n, k) array`*) – Data points matrix (data points = row vectors in the matrix)
- **`nb_eig`** (*`int, optional`*) – Number of leading principal components returned
- **`return_matrix`** (*`bool, optional`*) – If True, returns the matrix of the data points projection on the eigenvectors
- **`return_eigenvalues`** (*`bool, optional`*) – Returns the eigenvalues.

Returns

- *(k, nb_eig) array* – Leading principal components/eigenvectors (columnwise).
- **Proj** (*(t_max, N_G) array*) – If return_matrix == True: Projection of the data points on the principal eigenvectors.

`chaotic_neural_networks.utils.`**`add_collection_curves`**(*ax*, *ts*, *data*, *labels=None*, *color='indigo'*, *y_lim=None*, *starting_points=None*, *Δ=None*)

Adds a collection of curves a matplotlib ax.

`chaotic_neural_networks.utils.`**`both`**(*f*, *g*)

Generates the function \(t (f(t), g(t))\)

`chaotic_neural_networks.utils.`**`triple`**(*f*, *g*, *h*)

Generates the function \(t (f(t), g(t), h(t))\)

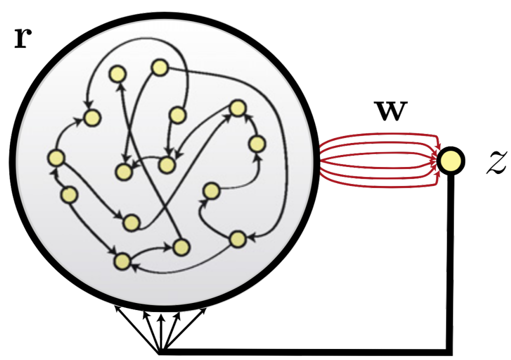
1.3 Network Architecture A

`chaotic_neural_networks.networkA.`**`N_G = 1000`**

Generator Network – Number of neurons

`class` `chaotic_neural_networks.networkA.`**`NetworkA`**(*N_G=1000*, *p_GG=0.1*, *g_GG=1.5*, *g_Gz=1.0*, *f=<numpy.lib.function_base.vectorize object>*, *dt=0.1*, *Δt=1.0*, *α=1.0*, *τ=10.0*, *seed=1*, *nb_outputs=1*)

Neural Architecture A:



- A recurrent generator network with firing rates \(\mathbf{r}\) driving a linear readout unit with output \(z\) through weights \(\mathbf{r}\) that are modified during training.
- Feedback to the generator network is provided by the readout unit.

`FORCE_sequence`(*t_max*, *number_neurons=5*)

Returns a matplotlib figure of a full FORCE training sequence, showing the evolution of:

- network ouput(s)
- `number_neurons` neurons membrane potential
- and the time-derivative of the readout vector \(\dot{\textbf{w}}\)

before training (spontaneous activity), throughout training, and after training (test phase): each one of these phases lasts `t_max/3`.

See `training_sequence_plots.py` in the github repository for further examples.

Examples

>> network = networkA.NetworkA(f=utils.periodic); network.FORCE_sequence(600) Pre-training / Spontaneous activity... Training... > **Average Train Error:** [0.02805716] Testing... > **Average Test Error:** [2.50709125]

error (*train_test='train'*)

Compute the average training/testing error.

Parameters train_test (*{'PCA', 'MDA'}, optional*) – Choice of the error to compute: train or test.

Returns Train of test error, depending on *train_test*

Return type (len(self.z_list),) array

step (*train_test='train'*, *store=True*)

Execute one time step of length `dt` of the network dynamics.

Parameters train_test (*{'PCA', 'MDA'}, optional*) – Learning phase (when \(P\) and the readout unit are updated) or test phase (no such update)

Examples

```
>>> from chaotic_neural_networks import networkA; net = networkA.NetworkA()
>>> for _ in np.arange(0, 1200, net.dt):
...     net.step()
>>> net.error()
0.015584795078446064
```

chaotic_neural_networks.networkA.**dt = 0.1**

Network integration time step.

chaotic_neural_networks.networkA.**g_GG = 1.5**

Scaling factor of the connection synaptic strength matrix of the generator network. $$g_{GG} > 1 ext{chaotic behavior}$$

chaotic_neural_networks.networkA.**g_Gz = 1.0**

Scaling factor of the feedback loop – Increasing the feedback connections result in the network chaotic activity allowing the learning process.

chaotic_neural_networks.networkA.**p_GG = 0.1**

Generator Network – **sparseness parameter** of the connection matrix. Each coefficient thereof is set to \(0\) with probability \(1-p_{GG}\).

chaotic_neural_networks.networkA.**p_z = 1.0**

Sparseness parameter of the readout – a random fraction \(1-p_z\) of the components of \(\mathbf{w}\) are held to \(0\).

chaotic_neural_networks.networkA.**Δt = 1.0**

Time span between modifications of the readout weights – \(Δt dt\)

chaotic_neural_networks.networkA.α **= 1.0**

Inverse Learning rate parameter – \(P\), the estimate of the inverse of the network rates correlation matrix plus a regularization term, is initialized as $$P(0) = \frac 1 α \mathbf{I}$$

So a sensible value of \(α\) - depends on the target function - ought to be chosen such that \(α << N\)

If - \(α\) is too small the learning is so fast it can cause unstability issues. - \(α\) is too large the learning is so slow it may fail `chaotic_neural_networks.networkA.τ = 10.0`

Time constant of the units dynamics.

1.4 Indices and tables

- genindex
- modindex
- search

Python Module Index

c

Index