lapart Documentation

Release 0.0.1

C. Birk Jones

Mar 03, 2018

Overview:

The Laterally Primed Adaptive Resonance Theory (LAPART) neural networks couple two Fuzzy ART algorithms to create a mechanism for making predictions based on learned associations. The coupling of the two Fuzzy ARTs has a unique stability that allows the system to converge rapidly towards a clear solution. Additionally, it can perform logical inference and supervised learning similar to fuzzy ARTMAP.

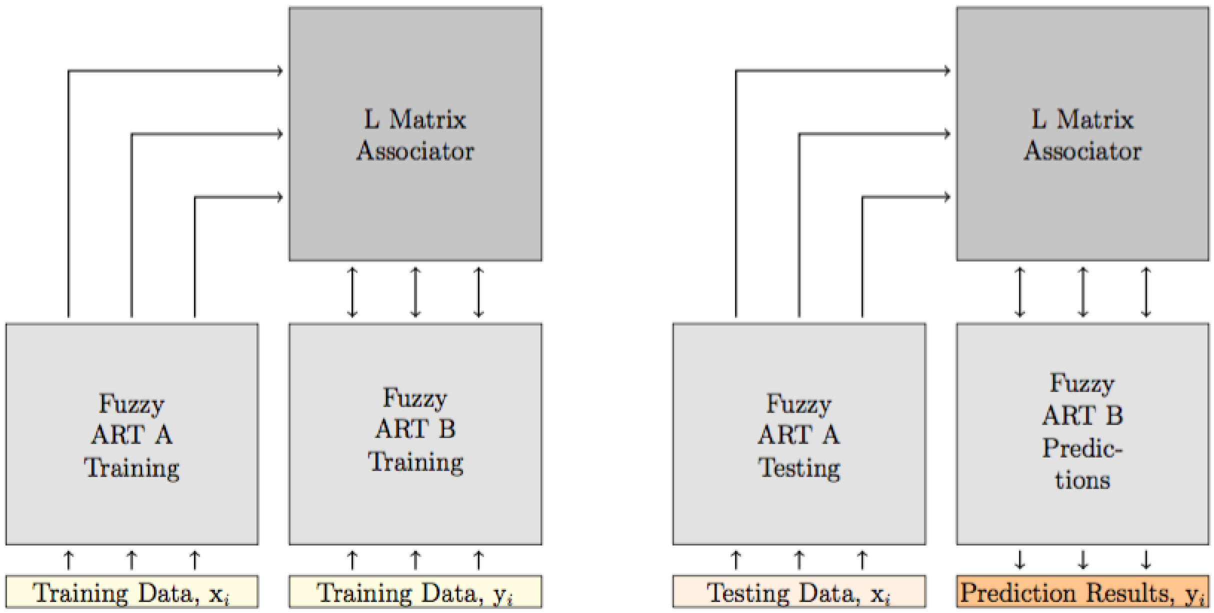


Figure 1: LAPART training (shown on the left side) uses two Fuzzy ART (A&B) algorithms connected by an associator matrix (L). During training inputs xi are applied to the A-Side while yi inputs are presented ot the B side. The algorithm then produces templates and an L matrix. The testing processes (shown on the right side) has the same structure as the training but applies previously unseen testing data (xi) to the A-Side. The algorithm then produces outputs on the B-Side that are the prediction results.

CHAPTER 1

Architecture

The general layout of the LAPART algorithm includes two fuzzy ARTs, labeled as A and B, that are connected by an associator matrix referred to as L. Each of them have an input layer, a recognition layer, and a categorization layer. Also, they both have a vigilance parameter rhoA, rhoB respectively. The A and B algorithms are connected together by an inference mechanism so that the template connections are established during training and then used to provide predictions during testing. The flow of the algorithm is shown in the Figure below. During training, the system is able to learn through the presentation of input pattern pairs (IA{x1..xn} and IB{y1..yn}) applied to each Fuzzy ART network. At the same time, interconnections between classes are formed in the L matrix. The interconnections between the A and B Fuzzy ART connect the learned categories and allow for predictions to be made in the testing phase when new data becomes available. During testing previously unseen data are presented to the A side only. Categorization of the input patterns occurs in the A side which connects, through the L matrix, to a particular category on the B side. The particular B side category for the input pattern is then the prediction for the given A input.

1.1 Fuzzy ART A

The training process is initiated with the presentation of an input pattern into the Fuzzy ART A algorithm. Since no templates exist at the onset of the training the initial input becomes the first template on the A-side. Therefore, a new template or class would be created using the "New A Class" block shown in the figure above. In this situation, the Fuzzy ART B operates as a normal ART algorithm. Since there are no B-Side templates, a new template was automatically created and a link between the A and B Fuzzy ARTs are established in the L matrix.

The algorithm then starts over with a new input, and repeats until all of the inputs are considered. When the new input is presented to all of the existing templates, in the "A-Side Search" block, it searched for the top matches as determined by the choice function. Then the vigilance test is computed in the "A-Side Resonance" block to determine if resonance with an existing template occurs or not. If resonance does not occur then the next best template was considered until no choices were left. If a match is not found, then the script for Case 1 is implemented. If a match does occur, the script for Case 2 is used.

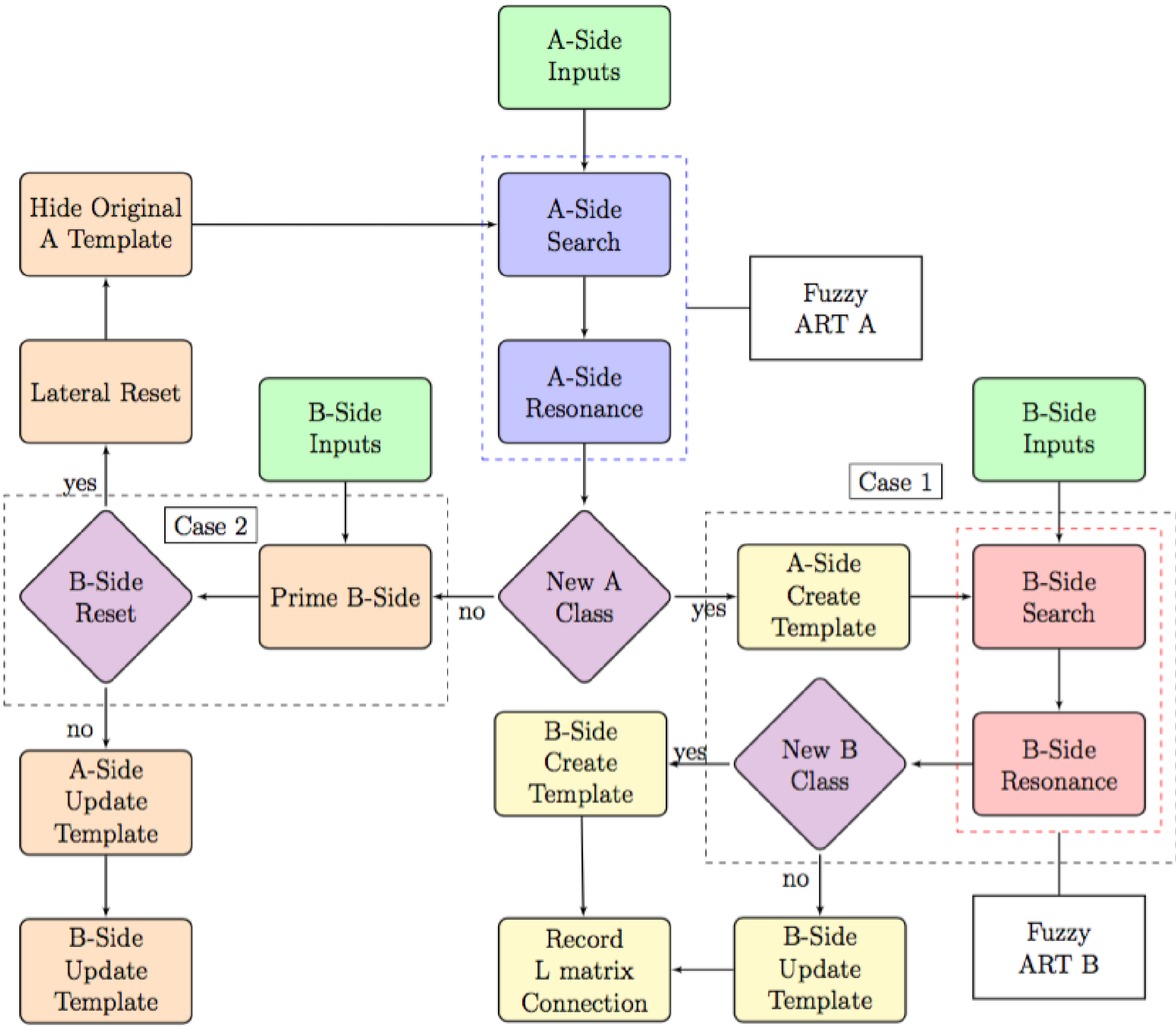


Figure 1.1: LAPART training algorithm flow diagram includes Fuzzy ART A & B and two cases for learning A and B side templates.

1.2 Case 1

The Case 1 code scenario occurs when a new A-side class was created and the Fuzzy ART B is allowed to operate as a normal ART algorithm. First, a new A-Side template is created. Then the B-Side input pattern is presented to the "B-Side Search" block where the choice function is used to discover the best template match. After the best match is found, the system tested the match to see if it met the vigilance parameter criteria in the "B-Side Resonance" block. If it did not, then a new template is created. If it did, then it updates an existing template. After the creation or update of a B-Side template, the inference matrix L was updated to link the newly created A-Side template to the B-Side template.

1.3 Case 2

In the event that resonance occurred in the Fuzzy ART A section of the code then Case 2 is implemented. First, the A-side template that resonated with the input pattern is not updated, but instead put on hold until further notice. Also, the match function is used to find the template that best matched the given input pattern. Then, if the chosen template passed the vigilance criteria, where the match function was greater than or equal to the B-side vigilance (rhoB), then the given A and B side templates are updated respectively. But, if it does not pass, then the system experiences a lateral reset and the initial A-side template is hidden and the process is repeated.

CHAPTER 2

Installation

lapart-python requires Python (2.7, 3.4, or 3.5) along with several Python package dependencies. Information on installing and using Python can be found at https://www.python.org/. Python distributions, such as Anaconda, are recommended to manage the Python interface.

lapart-python can be installed using pip, git, or a downloaded zip file. Note that the pip installation will NOT include the examples folder referenced in this manual.

pip: To install lapart-python using pip:

```
pip install lapart-python
```

Required Python package dependencies include:

- Pandas [Mcki13]: used to analyze and store time series data, http://pandas.pydata.org/
- Numpy [VaCV11]: used to support large, multi-dimensional arrays and matrices, http://www.numpy.org/

CHAPTER 3

Example: XOR

The exclusive or (XOR) is a logical operation that outputs true when inputs differ.

```
import numpy as np
import pandas as pd
from lapart import train,test
```

```
xtrain = pd.read_csv('xor_train.csv').as_matrix()
xAtest = pd.read_csv('xor_test.csv').as_matrix()
xAtrain,xBtrain = xtrain[:,0:2],xtrain[:,2:3]
```

```
xAtrain
```

```
array([[ 0. ,  0. ],
       [ 1. ,  0. ],
       [ 0. ,  1. ],
       [ 1. ,  1. ],
       [ 1. ,  1. ],
       [ 0.9,  0.9],
       [ 0.1,  0.8],
       [ 0.2,  0.2],
       [ 1. ,  1. ]])
```

```
xBtrain
```

```
array([[ 0. ],
       [ 1. ],
       [ 1. ],
       [ 0. ],
       [ 0. ],
       [ 0.1],
       [ 0.8],
       [ 0. ],
       [ 0. ]])
```

```
xAtest
```

```
array([[ 0.1 ,  0.9 ],
       [ 1.  ,  0.  ],
       [ 0.  ,  0.  ],
       [ 1.  ,  1.  ],
       [ 0.  ,  1.  ],
       [ 0.15,  0.1 ]])
```

```
rA,rB = 0.8,0.8
```

```
TA,TB,L,t = train.lapArt_train(xAtrain,xBtrain,rhoA=rA,rhoB=rB,memory_folder=
↪'templates',update_templates=False)
```

```
C,T,Tn,df,t = test.lapArt_test(xAtest,rhoA=rA,rhoB=rB,memory_folder='templates')
```

```
C
```

```
array([[ 1.],
       [ 1.],
       [ 0.],
       [ 0.],
       [ 1.],
       [ 0.]])
```

CHAPTER 4

Example: Solar PV

The LAPART algorithm has been used in solar photovoltaic (PV) applications: (http://ieeexplore.ieee.org/document/7355834/).

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import matplotlib.dates as mdates
```

```
from lapart import train,test
```

```
df = pd.read_csv('pv_train.csv')
df = df.set_index('datetime')
df.index = pd.to_datetime(df.index)
df = df[df['POAIrrad1_Avg'] > 0]
```

```
strain,etrain = '2017-03-01 00:00:00','2017-03-28 23:59:00'
stest,etest = '2017-03-29 07:00:00','2017-03-31 18:59:00'
dftrain = df[(df.index >= strain) & (df.index <= etrain)]
dftest = df[(df.index >= stest) & (df.index <= etest)]
```

```
dftrain = df.sample(frac=0.5)
```

```
xAtrain = np.array([dftrain['POAIrrad1_Avg'].tolist()]).T  # Plane of Array Irradiance
xBtrain = np.array([dftrain['Sys1Wdc_Avg'].tolist()]).T    # System 1 DC Power
xAtest = np.array([dftest['POAIrrad1_Avg'].tolist()]).T    # Plane of Array Irradiance
```

```
rA,rB = 0.97,0.98
```

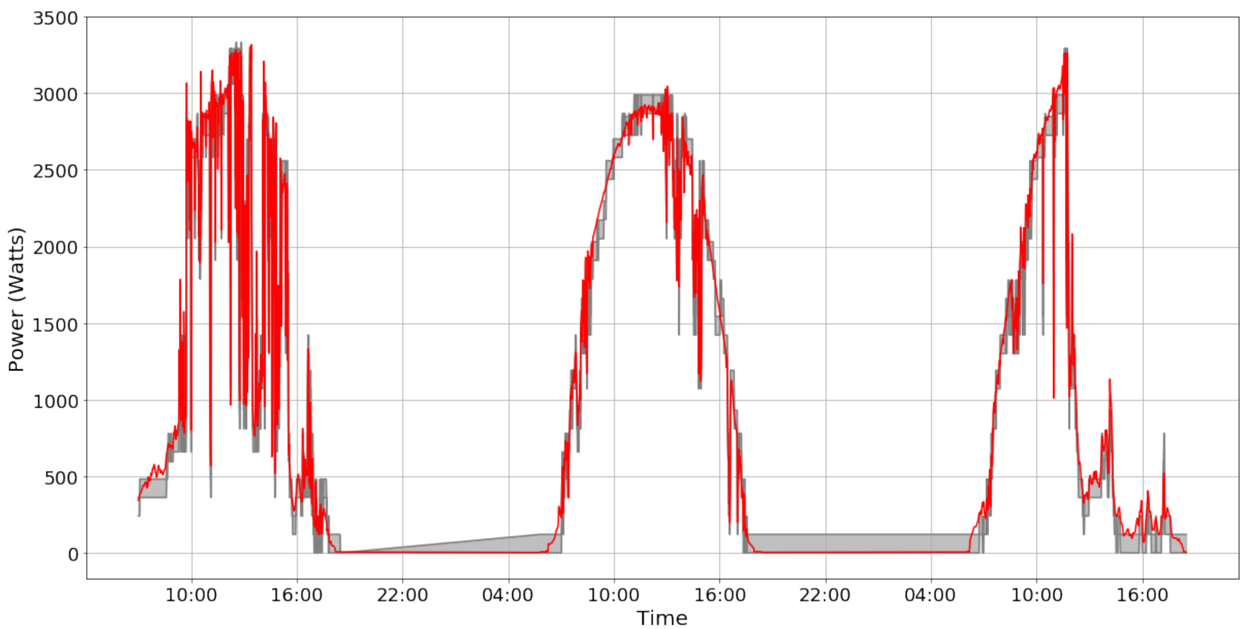
```
TA,TB,L,time_train = train.lapArt_train(xAtrain,xBtrain,rhoA=rA,rhoB=rB,memory_folder=
↪'templates',update_templates=False)
```

```
C,T,Tn,df,time_test = test.lapArt_test(xAtest,rhoA=rA,rhoB=rB,memory_folder='templates
↪')
```

```
dfn = pd.DataFrame(Tn,columns=['low', 'high'])
```

```
dftest['low'] = Tn[:,0].tolist()
dftest['high'] = Tn[:,1].tolist()
```

```
fig, (ax1) = plt.subplots(1,1,figsize=(20, 10))
ax1.plot(dftest['low'],color='grey')
ax1.plot(dftest['high'],color='grey')
ax1.fill_between(dftest.index, dftest['low'], dftest['high'], alpha=0.5,color='grey')
ax1.plot(dftest.index,dftest['Sys1Wdc_Avg'],color='red')
ax1.set_xlabel('Time',fontsize=20)
ax1.set_ylabel('Power (Watts)',fontsize=20)
ax1.tick_params(axis = 'both', which = 'major', labelsize = 18)

ax1.xaxis.set_major_formatter(mdates.DateFormatter('%H:%M'))
ax1.grid()

plt.show()
```

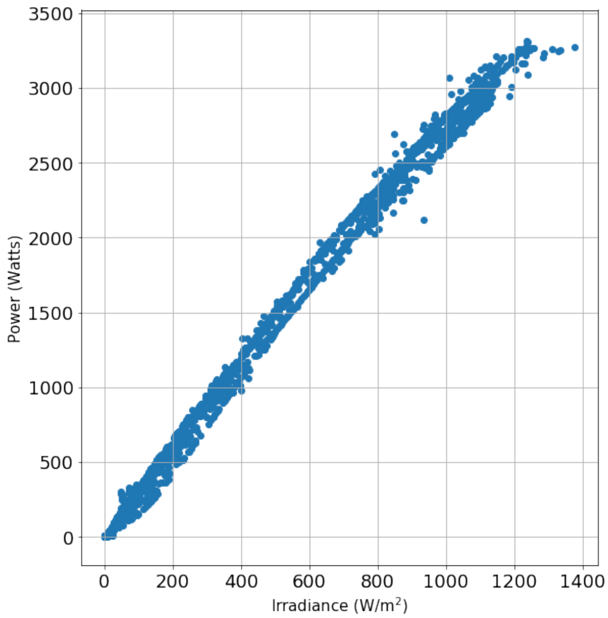


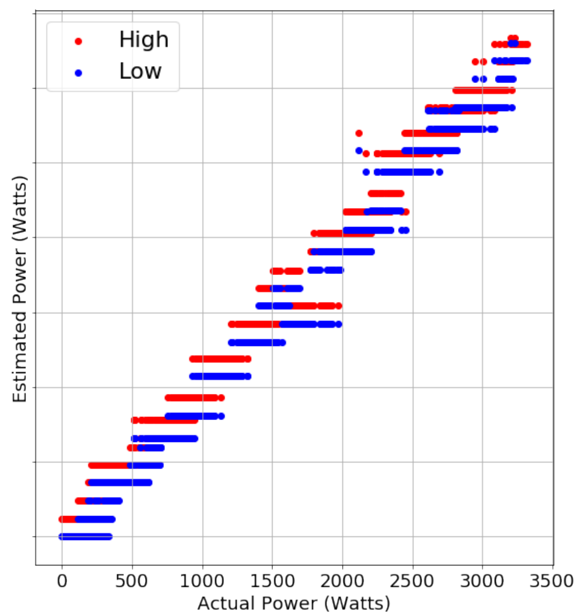

```
fig, (ax1,ax2) = plt.subplots(1,2,figsize=(20, 10),sharey=True)
ax1.scatter(dftest['POAIrrad1_Avg'],dftest['Sys1Wdc_Avg'])
ax1.set_xlabel('Irradiance (W/m$^2$)',fontsize=15)
ax1.set_ylabel('Power (Watts)',fontsize=15)
ax1.tick_params(axis = 'both', which = 'major', labelsize = 18)
ax1.grid()

ax2.scatter(dftest['Sys1Wdc_Avg'],dftest['high'],color='r')
ax2.scatter(dftest['Sys1Wdc_Avg'],dftest['low'],color='b')
ax2.set_xlabel('Actual Power (Watts)',fontsize=18)
ax2.set_ylabel('Estimated Power (Watts)',fontsize=18)
ax2.tick_params(axis = 'both', which = 'major', labelsize = 18)
```

(continues on next page)

(continued from previous page)

```
ax2.grid()


plt.show()
```





CHAPTER 5

Adaptive Resonance Theory

Fuzzy ART is a ANN architecture that can learn without forgetting. It is similar to human memory where people can recognize their parents even if they have not seen them in a while and have learned many new faces since. The theory was developed by Grossberg and Carpenter and includes various types such as ART 1, ART 2, ART 3, and Fuzzy ART. ART 1 is an architecture that can be used for clustering of binary inputs only. ART 2 improved upon the ART 1 architecture to support continuous inputs. Fuzzy ART, used in the present work, incorporates fuzzy set theory into the pattern recognition process.

The stand alone ART algorithm is available in Python at https://github.com/cbirkj/art-python.

`art.`**`match_choice`**(*c*, *norm*, *normI*, *normT*, *m*, *chm*, *rho*, *beta*)
Checks match criterion Compute choice equation Discovers best choice

Parameters

- **`norm`** – minimum of input and templates
- **`normI`** – norm of input

Returns returns category choice location

`art.`**`template_options_loop`**(*cmax*, *chmax*, *ch*, *nc*, *m*, *chm*, *rho*)
Match Criterion

Parameters

- **`cmax`** – Maximum choice (initialized to be -1)
- **`chmax`** – Match Criterion (initialized to be -1)
- **`ch`** – Template choice
- **`nc`** – Number of Categories

Return cmax Maximum choice template location

`art.`**`ART`**(*I*, *T*, *m*, *chm*, *nc*, *min_calc*, *rho*, *beta*, *j*)
Train ART - Create Template Matrix

Parameters

- **I** – Input
- **T** – Template
- **cmax** – Maximum choice (initialized to be -1)
- **chmax** – Match Criterion (initialized to be -1)

CHAPTER 6

Training Algorithm

The training code evaluates each input using :func:'lapart.train.lapart_train'

The supervised training algorithm used known inputs and outputs, and free parameters to create templates TA, TB, and L.

`train.`**`lapArt_train`**(*xA*, *xB*, *rhoA=0.9*, *rhoB=0.9*, *beta=1e-06*, *alpha=1.0*, *nep=1*, *memory_folder=''*, *update_templates=True*, *normalize_data=True*)

Train LAPART Algorithm

Parameters

- **`xA`** – A-Side Input Matrix (float)
- **`xB`** – B-Side Input Matrix (float)
- **`rhoA`** – A-Side free parameter (float)
- **`rhoB`** – B-Side free parameter (float)
- **`beta`** – Learning rate free parameter (float)
- **`alpha`** – Choice Parameter (float)
- **`nep`** – Number of epochs (integer)
- **`memory_folder`** – Folder to store memory (string)
- **`update_templates`** – Command to update or create new templates (boolean)

Return TA A-Side template matrix (float)

Return TB B-Side template matrix (float)

Return L Associator matrix (float)

Return elapsed_time Seconds to complete training (float)

CHAPTER 7

Testing Algorithm

The testing algorithm uses the stored memory (TA, TB, and L) to make predictions.

CHAPTER 8

Copyright and license

lapart-python is copyright through Sandia National Laboratories. The software is distributed under the Revised BSD License. Pecos also leverages a variety of third-party software packages, which have separate licensing policies.

8.1 Copyright

8.2 Revised BSD license

(continues on next page)

(continued from previous page)

CHAPTER 9

Indices and tables

- genindex
- modindex
- search

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology and Engineering Solutions of Sandia, LLC., a wholly owned subsidiary of Honeywell International, Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA-0003525.

Python Module Index

Index