LACE Documentation

Release 0+untagged.88.g06d5f4f.dirty

Jianfeng Chen

May 17, 2018

Contents

Contents:

CHAPTER 1

LACE

- Free software: MIT license
- Documentation: http://lace.readthedocs.io/en/latest/readme.html
- Algorithm design: Dr. Fayola Peters @ Univ of Limerick, Ireland
- Package development: Jianfeng Chen @ NC State Univ, United States

1.1 What is LACE?

LACE, or Large-scale Assurance Configuration Environment, was firsed introduced by Dr. Peters in ICSE2013. In a short, LACE is a data preprocess algorithm. It can help user to remove the sensitive information and implicit association rules inside the date sets, while keep the utility of the data sets, typically for machine learning or big data mining. In our published articiles, we used the data to train learning models and do the prediction.

There are two versions of LACE at this time. The first version, or *lace1* is constructed by two parts– CLIFF and MORPH. *CLIFF* is to find the most valuable subset among the dataset. *MORPH* is to "shake" the data so that someone else can not reveal the original data and remove the implicit association rules among the attributes.

The second version of LACE, or lace2, assumes there is a bin which contains the privatized data set from other people or insititutions. And lace2 can allow the user to determine what he or she should add to the bin so that it can improve the diversity of the bin. To pratitize the data, MORPH is also used in lace2.

To explore more details of the lace1 and lace2, please see the two papers listed in **Bibtex**.

1.2 How to use?

LACE can be easily installed by *pip*. Check **Installation** and **Usage**.

1.3 Bibtex

```
@inproceedings{peters2015lace2,
  title={LACE2: better privacy-preserving data sharing for cross project defect␣
↪prediction},
  author={Peters, Fayola and Menzies, Tim and Layman, Lucas},
  booktitle={Proceedings of the 37th International Conference on Software Engineering-
↪Volume 1},
  pages={801--811},
  year={2015},
  organization={IEEE Press}
}
```

```
@article{peters2013balancing,
  title={Balancing privacy and utility in cross-company defect prediction},
  author={Peters, Fayola and Menzies, Tim and Gong, Liang and Zhang, Hongyu},
  journal={IEEE Transactions on Software Engineering},
  volume={39},
  number={8},
  pages={1054--1068},
  year={2013},
  publisher={IEEE}
}
```

CHAPTER 2

Installation

At the command line:

```
$ pip install git+git://github.com/Ginfung/LACE
```

Just a kind reminder, the package may be installed into some other folder, which is not included in PYTHONPATH, such as /usr/local/lib/python2.7/site-packages/ . Python can only recongnize the packages in PYTHONPATH or the current working directory. If you come across this situation, please check http://stackoverflow.com/questions/12311085/how-to-permanently-append-a-directory-to-pythonpath .

If you don't know where LACE is installed, just run "pip uninstall lace" . You will get the answer.

CHAPTER 3

Usage

To use LACE in a project:

```
import lace # or
from lace import cliff, morph, lace1, add_to_bin, lace2_simulator
```

The CLIFF func:

```
cliff(attribute_names, data_matrix, independent_attrs, objective_attr, objective_as_
↪binary=False, cliff_percentage=0.4)
```

param attribute_names the attribute names. This should match the data_matrix

param data_matrix the data to trim

param independent_attrs set up the independent attributes in the dataset. Note: 'name', 'id', etc. might not be considered as independent attributes

param objective_attr marking which attribute is the objective to be considered

param objective_as_binary signal to set up whether treat the objective as a binary attribute. Default: False

param cliff_percentage set up how many records to be remained. By default, it is 0.4

return the survived (valued) records

The MORPH func:

```
morph(attribute_names, data_matrix, independent_attrs, objective_attr, objective_as_
↪binary=False, data_has_normalized=False, alpha=0.15, beta=0.35)
```

param attribute_names the names of attributes, should match the data_matrix

param data_matrix original data

param independent_attrs set up the independent attributes in the dataset. Note: 'name', 'id', etc. might not be considered as independent attributes

param objective_attr marking which attribute is the objective to be considered

param objective_as_binary signal to set up whether treat the objective as a binary attribute. Default: False

param data_has_normalized telling whether the data matrix has been normalized.

param alpha morph algorithm parameter

param beta morph algorithm parameter

return handled records

The most convenient way to use LACE1 is:

```
lace1(attribute_names, data_matrix, independent_attrs, objective_attr, objective_as_
↪binary=False, cliff_percentage=0.4, alpha=0.15, beta=0.35)
```

param attribute_names the names of attributes, should match the data_matrix

param data_matrix original data

param independent_attrs set up the independent attributes in the dataset. Note: 'name', 'id', etc. might not be considered as independent attributes

param objective_attr marking which attribute is the objective to be considered

param objective_as_binary signal to set up whether treat the objective as a binary attribute. Default: False

param cliff_percentage prune rate

param alpha parameter 1 in morph, defining the shaking degree

param beta parameter 2 in morph, defining the shaking degree

The data selection and processor in LACE2:

```
add_to_bin(attribute_names, try2add_data_matrix, independent_attrs, objective_attr,␣
↪objective_as_binary=False, cliff_percentage=0.4, morph_alpha=0.15, morph_beta=0.35,␣
↪passing_bin=None)
```

param attribute_names the names of attributes, should match the data_matrix

param try2add_data_matrix the data anyone is holding

param independent_attrs set up the independent attributes in the dataset. Note: 'name', 'id', etc. might not be considered as independent attributes

param objective_attr marking which attribute is the objective to be considered

param objective_as_binary signal to set up whether treat the objective as a binary attribute. Default: False

param cliff_percentage prune rate

param morph_alpha parameter 1 in morph, defining the shaking degree

param morph_beta parameter 2 in morph, defining the shaking degree

param passing_bin the data get from someone else. Set None if no passing data

return the new passing_bin. NOTE: the result must be assigned to another variable. The parameter pointer will NOT be changed

LACE also provides a simple LACE2 application simulator. It automatically distribute all data to different members UNEQUALLY.:

```
lace2_simulator(attribute_names, data_matrix, independent_attrs, objective_attr,␣
↪objective_as_binary=False, cliff_percentage=0.4, morph_alpha=0.15, morph_beta=0.35,␣
↪number_of_holder=5)
```

Here we have a complete simple example to propess the data. *This data is from Data.Gov*

```
import lace
import csv

with open('example.csv', 'r') as f:
    reader = csv.reader(f)
    header = next(reader)
    data = list()
    for line in reader:
        data.append(line)

attribute_names = header
data_matrix = data
independent_attrs = ['ADM_RATE', 'SAT_AVG', 'TUITFTE', 'RET_FT4', 'PCTFLOAN', 'PCTPELL
↪', 'DEBT_MDN', 'C150_4', 'CDR3']
objective_attr = 'mn_earn_wne_p7'

aftercliff = lace.cliff(attribute_names, data_matrix, independent_attrs, objective_
↪attr, False, 0.4)
assert len(aftercliff) < 600

aftermorph = lace.morph(attribute_names, aftercliff, independent_attrs, objective_
↪attr, False, False, 0.15, 0.35)
assert len(aftermorph)==len(aftercliff) and aftermorph[0] != aftercliff[0]


lace1res = lace.lace1(attribute_names, data_matrix, independent_attrs, objective_attr,
↪ False, 0.4, 0.15,0.35)
assert len(lace1res) < len(data)*0.5

bins = [header] + data[:50]
try2add_data_matrix = data[200:700]
bins = lace.add_to_bin(attribute_names, try2add_data_matrix, independent_attrs,␣
↪objective_attr, False, 0.4, 0.15, 0.35, bins)
assert len(bins) < 550


lace2res = lace.lace2_simulator(attribute_names, data_matrix, independent_attrs,␣
↪objective_attr, False, 0.4, 0.15, 0.35, number_of_holder=5)
assert len(lace2res)<len(lace1res)
```

CHAPTER 4

Contributing

Contributions are welcome, and they are greatly appreciated! Every little bit helps, and credit will always be given.

You can contribute in many ways:

4.1 Types of Contributions

4.1.1 Report Bugs

Report bugs at https://github.com/Ginfung/LACE/issues.

If you are reporting a bug, please include:

- Any details about your local setup that might be helpful in troubleshooting.
- Detailed steps to reproduce the bug.

4.1.2 Fix Bugs

Look through the GitHub issues for bugs. Anything tagged with "bug" is open to whoever wants to implement it.

4.1.3 Implement Features

Look through the GitHub issues for features. Anything tagged with "feature" is open to whoever wants to implement it.

4.1.4 Write Documentation

LACE could always use more documentation, whether as part of the official LACE docs, in docstrings, or even on the web in blog posts, articles, and such.

4.1.5 Submit Feedback

The best way to send feedback is to file an issue at https://github.com/Ginfung/LACE/issues.

If you are proposing a feature:

- Explain in detail how it would work.
- Keep the scope as narrow as possible, to make it easier to implement.
- Remember that this is a volunteer-driven project, and that contributions are welcome :)

4.2 Get Started!

Ready to contribute? Here's how to set up *LACE* for local development.

1. Fork the *LACE* repo on GitHub.
2. Clone your fork locally:

```
$ git clone https://github.com/Ginfung/LACE.git
```

3. Install your local copy into a virtualenv. Assuming you have virtualenvwrapper installed, this is how you set up your fork for local development:

```
$ mkvirtualenv LACE
$ cd LACE/
$ python setup.py develop
```

4. Create a branch for local development:

```
$ git checkout -b name-of-your-bugfix-or-feature
```

   Now you can make your changes locally.

5. When you're done making changes, check that your changes pass flake8 and the tests, including testing other Python versions with tox:

```
$ flake8 lace tests
$ python setup.py test
$ tox
```

   To get flake8 and tox, just pip install them into your virtualenv.

6. Commit your changes and push your branch to GitHub:

```
$ git add .
$ git commit -m "Your detailed description of your changes."
$ git push origin name-of-your-bugfix-or-feature
```

7. Submit a pull request through the GitHub website.

4.3 Pull Request Guidelines

Before you submit a pull request, check that it meets these guidelines:

1. The pull request should include tests.

2. If the pull request adds functionality, the docs should be updated. Put your new functionality into a function with a docstring, and add the feature to the list in README.rst.
3. The pull request should work for Python 2.7, 3.3, 3.4, 3.5 and for PyPy. Check https://travis-ci.org/Ginfung/LACE/pull_requests and make sure that the tests pass for all supported Python versions.

CHAPTER 5

Credits

5.1 Maintainer

- Jianfeng Chen <jchen37@ncsu.edu>

5.2 Contributors

None yet. Why not be the first? See: CONTRIBUTING.rst

CHAPTER 6

History

Pre-release

CHAPTER 7

Indices and tables

- genindex
- modindex
- search