Feature Vector Matrix Documentation

Release 0.1.1

Jeremy Robin

September 15, 2014

Contents

Contents:

CHAPTER 1

Feature Vector Matrix

Python class to encapsulate different representations of large datasets

- Free software: BSD license
- Documentation: https://featurevectormatrix.readthedocs.org.

1.1 Features

- Python class to encapsulate different representations of large datasets
- Rows can be inserted as python dictionaries or arrays
- They can be pulled out as dictionaries or arrays

CHAPTER 2

Installation

At the command line:

```
$ easy_install featurevectormatrix
```

Or, if you have virtualenvwrapper installed:

```
$ mkvirtualenv featurevectormatrix
$ pip install featurevectormatrix
```

CHAPTER 3

Usage

To use Feature Vector Matrix in a project:

```
import featurevectormatrix
```

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/talentpair/featurevectormatrix/issues.

If you are reporting a bug, please include:

- Your operating system name and version.
- 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

Feature Vector Matrix could always use more documentation, whether as part of the official Feature Vector Matrix 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/talentpair/featurevectormatrix/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 *featurevectormatrix* for local development.

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

   ```
   $ git clone git@github.com:your_name_here/featurevectormatrix.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 featurevectormatrix
   $ cd featurevectormatrix/
   $ 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 --config=flake8.cfg .
   $ 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. Check https://travis-ci.org/talentpair/featurevectormatrix/pull_requests and make sure that the tests pass for all supported Python versions.

4.4 Tips

To run a subset of tests:

```
$ python -m unittest tests.test_featurevectormatrix
```

CHAPTER 5

Credits

5.1 Development Lead

- Jeremy Robin <jeremy.robin@gmail.com>

5.2 Contributors

None yet. Why not be the first?

CHAPTER 6

History

CHAPTER 7

0.1.1 (2014-09-12)

- Improve import path

CHAPTER 8

0.1.0 (2014-09-12)

- First release on PyPI.

CHAPTER 9

featurevectormatrix

9.1 featurevectormatrix package

9.1.1 Module contents

class `featurevectormatrix.`**`FeatureVectorMatrix`**(*default_value=0*, *default_to_hashed_rows=False*, *rows=None*)

Bases: `object`

A class to abstract away the differences in internal representation between dictionaries and lists that can matter for very large datasets of vectors and allow them to work seamlessly with each other

Supports indexing and iteration (fvm[1] and for i in fvm:...) but you should set default_to_hash_rows to get the expected behavior. Also supports len

`add_row`(*list_or_dict*, *key=None*)

Adds a list or dict as a row in the FVM data structure

Parameters

- **key** (*str*) – key used when rows is a dict rather than an array
- **list_or_dict** – a feature list or dict

`column_count`()

Get the current number of columns

Returns the count

`column_names`()

get the column names

Returns The ordered list of column names

`default_to_hashed_rows`(*default=None*)

Gets the current setting with no parameters, sets it if a boolean is passed in

Parameters **default** – the value to set

Returns the current value, or new value if default is set to True or False

`extend_rows`(*list_or_dict*)

Add multiple rows at once

Parameters **list_or_dict** – a 2 dimensional structure for adding multiple rows at once

Returns

get_matrix()
Use numpy to create a real matrix object from the data

Returns the matrix representation of the fvm

get_row_dict(*row_idx*)
Return a dictionary representation for a matrix row

Parameters **row_idx** – which row

Returns a dict of feature keys/values, not including ones which are the default value

get_row_list(*row_idx*)
get a feature vector for the nth row

Parameters **row_idx** – which row

Returns a list of feature values, ordered by column_names

keys()
Returns all row keys

Raises **NotImplementedError** if all rows aren't keyed

Returns all row keys

row_count()
The current number of rows

Returns the count

row_names()
get the column names

Returns The ordered list of column names

set_column_names(*column_names*)
Setup the feature vector with some column names :param column_names: the column names we want :return:

set_row_names(*row_names*)
Setup the feature vector with some column names :param row_names: the column names we want :return:

transpose()
Create a matrix, transpose it, and then create a new FVM

Raises **NotImplementedError** if all existing rows aren't keyed

Returns a new FVM rotated from self

CHAPTER 10

Indices and tables

- *genindex*
- *modindex*
- *search*

f