

MOE: Mixture-of-Experts Software Manual

Harini Eavani

Harini.Eavani@uphs.upenn.edu

cbica-software@uphs.upenn.edu

March 17, 2016

1 Introduction

This software can be used for datasets where the reference group ("controls") and the affected group ("patients") cannot be separated by a single line ("hyperplane"). MOE combines multiple hyperplanes along with a clustering objective, to split the affected group into multiple sub-groups. This is done in such a manner that each of the resulting sub-group is separable from the reference group by a single line.

See Figure 1, where the reference group is denoted by circles, and the affected group by triangles. In Case1, the affected group is not heterogeneous, as it is separable from the reference group with a single line. Cases 2, 3, 4 require multiple lines to separate the affected group from the reference group.

This software works with data saved in comma-separated value format. Input features can be based on any type of data - volumes, density, connectivity, diffusion, clinical scores, cognitive data etc.

2 Method

Consider a binary classification problem with data $\mathbf{x}_i \in \mathbf{R}^D$ obtained from $i = 1, 2, \dots, N$ subjects. Each subject is associated with a binary label $y_i \in \{-1, 1\}$, -1 for the reference group and $+1$ for the affected group.

Let $\mathbf{m}_i = [m_i^1, m_i^2, \dots, m_i^K]$, $m_i \in [0, 1]$, $\sum_{k=1}^K m_i^k = 1$ indicate the relative membership of subject i to group k .

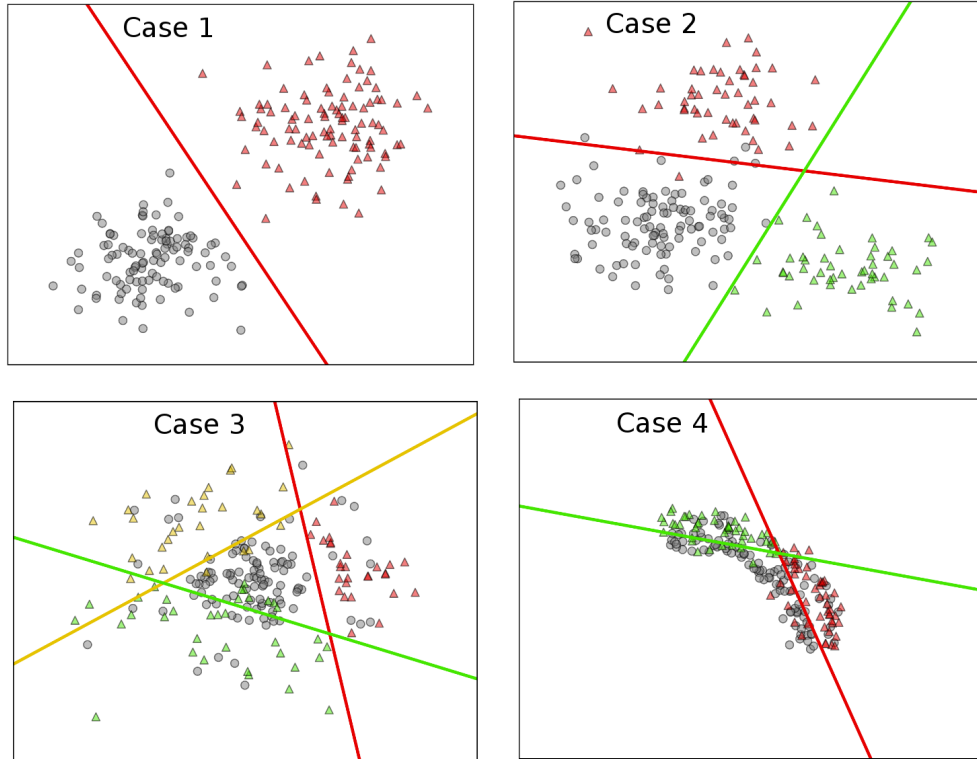


Figure 1: Four test cases with circles as the reference (controls) . Case 1 has no heterogeneity, as the reference and affected group (triangles) are separable by a single line. Case 2, 3, and 4 have different amounts of heterogeneity - MOE is able to find 2, 3 and 2 affected sub-groups respectively. These are indicated by different colors for the data points and associated hyperplanes.

MOE is a joint model that combines ℓ_1 -loss, ℓ_2 -regularized SVM and a Fuzzy-C Mean clustering objective. The optimization problem is given below.

$$\begin{aligned}
& \underset{\{\mathbf{w}^k\}_k, \{m_i^k\}_{i,k}}{\text{minimize}} && \sum_{k=1}^K \left\{ \frac{1}{2} \|\mathbf{w}^k\|_2 + C \sum_{i=1}^N m_i^k (1 - y_i (\mathbf{w}^k)^T \mathbf{x}_i)_+ + t \sum_{i=1}^N (m_i^k)^a \|\mathbf{x}_i - \mathbf{d}^k\|_F^2 \right\} \\
& \text{subject to} && \sum_{k=1}^K m_i^k = 1, \quad m_i^k \in [0, 1] \quad n = 1, \dots, N
\end{aligned} \tag{1}$$

where t is a user-defined value that controls the trade-off between the cost of classification and clustering. The other user-defined parameters are the number of groups K , and the SVM cost-value C .

The “fuzzyness ” coefficient m controls the “softness” of the membership assignments. In our model, we set the “fuzziness” coefficient a to a value of 2.

We use alternating minimization to solve for the cluster centroids \mathbf{d}^k , membership values m_i^k and the SVM hyperplanes \mathbf{w}^k . We use lib-svm with weighted samples to solve each of the \mathbf{w}^k , a quadratic-program to solve for m_i^k and the centroids have an analytical solution.

Given a new test case, its label y^* is determined as the sign of the weighted (based on its membership to each of the clusters) combination of individual SVM predictions.

Regression The regressed version uses Ridge regression in place of linear SVM. However please note that this extension of the code has not been extensively tested, nor published.

3 Software

3.1 Within main directory

1. Detailed documentation is contained in the latex file [UserManual.tex](#) and the pdf [UserManual.pdf](#) produced from it.
2. Author information is in [AUTHORS.txt](#)
3. All information needed to get started with the software is in [README.md](#)

4. The file `INSTALL.txt` has installation and usage information.
5. `Doxyfile` is the configuration file for generating doxygen documentation.
6. `ignore.txt` contains a list of files that doxygen ignores from parsing.
7. The folder `docs/` will contain all the documentation once doxygen is run using `Doxyfile`.
8. The folder `src/` contains source code for MOE.
9. The folder `licenses/` contains license information related to this software.

3.2 Within `src/` directory

This software is implemented entirely in Python.

1. Main python script is `moe`. This function parses input arguments, and calls `SV_fuzzy` to run the method. It also reads and saves output csv files. It calls `RepeatedKFold` if needed, if the `-f` argument is > 0 .
2. `RepeatedKFold.py` Repeated evaluation of K fold cross-validation.
3. `SV_fuzzy.py` The main function that implements the method. It takes as input data matrices in numpy format, and returns output also in numpy format. It calls `QP.py` for solving membership values.
4. `QP.py` It sets up the quadratic program for solving membership values.
5. `test.py` This script runs all four test cases, and calls `draw_moe_plot.py` to plot results.
6. `draw_moe_plot.py` This script plots the output for the two-dimensional test cases.
7. `MOEUtils.py` Contains set of generic functions for checking file existence, exit status, etc.
8. The python script `make` copies all python files to correct places in the install directory.

4 Testing and Installation

4.1 Dependencies

This software has been primarily implemented for Linux operating systems.

- Python 2.7.9
- Python library numpy 1.7.2
- Python library scipy 0.15.1
- Python library pandas 0.16.2
- Python library sklearn 0.15
- Python library matplotlib 1.4.3

Make sure all dependencies are met before proceeding with install.

4.2 Testing

Run `./make test` from within the `src` directory.

This runs `test.py` on four test cases.

To test the software, we have provided synthetic two dimensional data for four test cases. In each of these cases, data points denoted by circles are the reference (control) samples, while points denoted by triangles are the affected (patient) cases which are made up of heterogeneous sub-groups. These are located in `src/test/Case*`. The image `test/All_test_results.png` shows the four test cases, it is also shown in Figure 1.

Test cases are used to check if the cross-validation accuracy in four test cases is greater than that of linear-SVM.

Proceed to the install step only if test passes in all four cases.

4.3 Installation

Run `./make install` from within the `src` directory, with the location of the install directory as the argument:

```
make install <installDir>
```

make creates four folders within `installDir` - `bin`, `libexec`, `share` and `doc`.

1. bin/ contains the main executable moe.
2. libexec/ has all python scripts.
3. share/ has the test data.
4. doc/ has the doxygen documentation.

Add the install directory to your path by running the following command.
 Replace \$installDir with the location of the install directory from step 1 above.

```
export PATH=$PATH:$installDir/bin/
```

4.4 Code documentation

The ”./make install” step (above) generates documentation in \$installDir/doc/.*

5 Usage

```
moe --
Runs MOE-based heterogeneity classification on spreadsheet data
Works on any set of features -
Average regional volume/density/diffusion/connectivity,
clinical scores, cognitive data, etc
```

For best performance, pick optimal -c and -t values based on cross-validation

Usage: moe [OPTIONS]

Required Options:

[-d --data]	Specify the spreadsheet with list of ROIs
[-p --prefix]	Specify the prefix of the output file
[-H --header]	Specify the header column that has the discrete labels
[-l --label]	Specify the label for the heterogenous group

Options:

[-n --nSVMs]	Specify the number of SVMs as a number. Default = 2.
[-c --cost]	Specify the SVM cost value. Default = 1.
[-t --tradeoff]	Specify the classification vs. clustering tradeoff.

Default = 0.1.

`[-f --folds]` Specify the number of cross-validation folds to run.
Default = 0. (not run)

`[-o --outputDir]` The output directory to write the results.
Defaults to the location of the input file

`[-w --workingDir]` Specify a working directory.
By default a tmp dir is created and used

`[-u --usage | -h --help]` Display this message

`[-v --verbose]` Verbose output

`[-V --Version]` Display version information

To run this software you will need:

1. An input csv file, with the following mandatory fields:
 - (a) Subject index in the first column
 - (b) Header row containing feature names in the first row
 - (c) A column containing the binary group labels - the name of the column and the value of the patient group (of whose heterogeneous sub-groups you want to find) needs to be provided in the command line options.

For a csv file `ROI_values.csv` that looks like below:

```
Subject_id, ROI_1, ROI_2, ROI_3, ..., Label, ...
BLSA_0001, 0.234, 0.4545, 0.212, ..., 1 , ...
BLSA_0002, 0.122, 0.1213, 0.3434, ..., -1, ...
...
...
```

Run the following command:

```
moe -d ROI_values.csv -p ROI_results -l 1 -n 3 -o ./ -v
```

In the output directory, the software returns:

1. One python numpy file containing all the results - `<prefix>_results_allResults.npz`
2. A csv file containing the hyperplanes, indexed against feature name in the first row. Filename `<prefix>_results_hyperplanes.csv`

3. A csv file containing the memberships, indexed against subject id in the first column. Filename `<prefix>_results_memberships.csv`

6 Citation

If you find this code useful, please cite:

Eavani, H., Hsieh, M. K., An, Y., Erus, G., Beason-Held, L., Resnick, S., Davatzikos, C. (2016). Capturing heterogeneous group differences using mixture-of-experts: Application to a study of aging. *NeuroImage*, 125, 498-514.