epage Documentation

Liguo Wang

Jul 10, 2020

Overview

CHAPTER 1

About the package

The epage (Evaluate Protein Activity with Gene Expression) package contains several programs to calculate the *composite expression score*, build and evaluate SVM model, and use SVM model to predict new cases.

Name	Description
gComposite.py	Calculates these Composite Expression Scores
SVM_performance.py	Calculates these performance metrics of K-fold cross-validation
SVM_predict.py	Build SVM model from “train_file” and then predict cases in “data_file”.
SVM_ROC.py	Plot Receiver operating characteristic (ROC) curves using K-fold cross-validation.

CHAPTER 2

Workflow

1. We define the downstream target genes of a particular transcription factor.
2. We collect the gene expression and mutation data. Use “Normal” and “Trundating” as training datasets.
3. We run the **gComposite.py** to generate composite expression scores
4. Run **SVM_performance.py** to check the performance of the SVM model, adjust training data and fine-tune the parameters. Usually, we need to run *SVM_performance.py* multiple times.
5. Run **SVM_ROC.py** to generate ROC curve to visualize the performance.
6. Run **SVM_predict.py** to predict new cases.

CHAPTER 3

Prerequisites

Note: You need to install these tools if they are not available from your computer.

- Python 3
- R
- R library GSVA (only required by **gComposite.py**)

CHAPTER 4

Python dependencies

- pandas
- numpy
- scipy
- sklearn

Note: You do NOT need to install these packages manually, as they will be automatically installed if you use pip3.

CHAPTER 5

Install

```
$ pip3 install epage
or
$ pip3 install git+https://github.com/liguowang/epage.git
```

CHAPTER 6

Upgrade

```
$ pip3 install epage --upgrade
```

CHAPTER 7

SVM_ROC.py

7.1 Description

Plot Receiver operating characteristic (ROC) curves using K-fold cross-validation.

Options:

--version show program's version number and exit

-h, --help show this help message and exit

-i INPUT_FILE, --input_file=INPUT_FILE Tab or space separated file. The first column contains *sample IDs*; the second column contains *sample labels* in integer (must be 0 or 1); the third column contains *sample label names* (string, must be consistent with column-2). The remaining columns contain featuers used to build SVM model.

-o OUT_FILE, --output=OUT_FILE The prefix of the output file.

-n N_FOLD, --nfold=N_FOLD The original sample is randomly partitioned into *n* equal sized subsamples (2 =< n <= 10). Of the n subsamples, a single subsample is retained as the validation data for testing the model, and the remaining n 1 subsamples are used as training data. default=5.

-C C_VALUE, --cvalue=C_VALUE C value. default=1.0

-s RAND_SEED, --seed=RAND_SEED random_state seed used by the random number generator. default=0

-k S_KERNEL, --kernel=S_KERNEL Specifies the kernel type to be used in the algorithm. It must be one of 'linear', 'poly', 'rbf', 'sigmoid', 'precomputed' or a callable. If none is given, 'rbf' will be used. default=linear

--xl=X_LOW The lower limit of X-axis (false positive rate). default=-0.05

--xu=X_UPPER The upper limit of X-axis (false positive rate). default=0.5

--yl=Y_LOW The lower limit of Y-axis (true positive rate). default=0.5

--yu=Y_UPPER The upper limit of Y-axis (true positive rate). default=1.05

7.2 Input files format

ID	Label	Label_name	feature_1	feature_2	feature_3	...	feature_n
sample_1	1	WT	1560	795	0.9716	...	feature_n
sample_2	1	WT	784	219	0.4087	...	feature_n
sample_3	1	WT	2661	2268	1.1691	...	feature_n
sample_4	0	Mut	643	198	0.5458	...	feature_n
sample_5	0	Mut	534	87	1.0545	...	feature_n
sample_6	0	Mut	332	75	0.5115	...	feature_n

7.3 Command

```
$ python3  SVM_ROC.py -i lung_CES_5features.tsv -o output_ROC -C 10
```

CHAPTER 8

SVM_performance.py

8.1 Description

Calculating performance metrics using K-fold cross-validation.

- F1_micro
- F1_macro
- Accuracy
- Precision
- Recall

8.2 Options

--version show program's version number and exit

-h, --help show this help message and exit

-i INPUT_FILE, --input_file=INPUT_FILE Tab or space separated file. The first column contains *sample IDs*; the second column contains *sample labels* in integer (must be 0 or 1); the third column contains *sample label names* (string, must be consistent with column-2). The remaining columns contain featuers used to build SVM model.

-n N_FOLD, --nfold=N_FOLD The original sample is randomly partitioned into *n* equal sized subsamples (2 =< n <= 10). Of the n subsamples, a single subsample is retained as the validation data for testing the model, and the remaining n 1 subsamples are used as training data. default=5.

-p N_THREAD, --nthread=N_THREAD Number of threads to use. default=2

-C C_VALUE, --cvalue=C_VALUE C value. default=1.0

-k S_KERNEL, --kernel=S_KERNEL Specifies the kernel type to be used in the algorithm. It must be one of 'linear', 'poly', 'rbf', 'sigmoid', 'precomputed' or a callable. If none is given, 'rbf' will be used. default=linear

8.3 Input files format

ID	Label	Label_name	feature_1	feature_2	feature_3	...	feature_n
sample_1	1	WT	1560	795	0.9716	...	feature_n
sample_2	1	WT	784	219	0.4087	...	feature_n
sample_3	1	WT	2661	2268	1.1691	...	feature_n
sample_4	0	Mut	643	198	0.5458	...	feature_n
sample_5	0	Mut	534	87	1.0545	...	feature_n
sample_6	0	Mut	332	75	0.5115	...	feature_n

8.4 Example of input file

```
$ cat lung_CES_5features.tsv
TCGA_ID Label   Group   gsva_p53_activated      gsva_p53_repressed      ssGSEA_p53_
↪activated    ssGSEA_p53_repressed    PC1
TCGA-22-4593-11A        0       Normal  0.97337963      -0.965872505    0.446594884
↪   -0.332230329    10.12036762
TCGA-22-4609-11A        0       Normal  0.974507532     -0.971830001    0.480743696
↪   -0.373937866    12.57932272
TCGA-22-5471-11A        0       Normal  0.981934732     -0.991054313    0.465087717
↪   -0.354705367    11.50908022
TCGA-22-5472-11A        0       Normal  0.914660832     -0.889643616    0.433541263
↪   -0.316566781    7.96785884
TCGA-22-5478-11A        0       Normal  0.983080513     -0.989789407    0.478239013
↪   -0.370840097    11.81998124
TCGA-22-5481-11A        0       Normal  0.958950969     -0.973021839    0.441116626
↪   -0.325822867    10.62201083
TCGA-22-5482-11A        0       Normal  0.97113164      -0.976324136    0.471515295
↪   -0.362373723    10.78576876
TCGA-22-5483-11A        0       Normal  0.957377049     -0.986013986    0.378674475
↪   -0.253223408    7.487083257
TCGA-22-5489-11A        0       Normal  0.963911525     -0.982725528    0.45219094
↪   -0.339061168    9.49806089
TCGA-22-5491-11A        0       Normal  0.981934732     -0.991054313    0.475345705
↪   -0.367218333    12.2813137
TCGA-33-4587-11A        0       Normal  0.90739615      -0.930774072    0.403446401
↪   -0.281428331    9.368460346
TCGA-33-6737-11A        0       Normal  0.962025316     -0.957522049    0.495340808
↪   -0.391557543    10.79155095
TCGA-34-7107-11A        0       Normal  0.949717514     -0.934120795    0.451010344
↪   -0.337452999    10.04177079
TCGA-34-8454-11A        0       Normal  0.992397661     -0.987269255    0.480060883
↪   -0.372603029    10.6050578
...
```

8.5 Command

```
$  python3  SVM_performance.py -i lung_CES_5features.tsv -C 10
```

Note: There is no rule of thumb to choose a C value, people can try a bunch of different C values and choose the one which gives you "best performance scores"

8.6 Output to screen

```
Preprocessing data ...
Evaluate metric(s) by cross-validation ...
F1 score is the weighted average of the precision and recall. F1 = 2 * (precision *␣
↪recall) / (precision + recall)


F1_macro calculate metrics for each label, and find their unweighted mean. This does␣
↪not take label imbalance into account.
       Iteration 1: 1.000000
       Iteration 2: 0.983518
       Iteration 3: 1.000000
       Iteration 4: 1.000000
       Iteration 5: 0.967273
F1-macro: 0.9902 (+/- 0.0262)


F1_micro calculate metrics globally by counting the total true positives, false␣
↪negatives and false positives.
       Iteration 1: 1.000000
       Iteration 2: 0.986301
       Iteration 3: 1.000000
       Iteration 4: 1.000000
       Iteration 5: 0.972222
F1-micro: 0.9917 (+/- 0.0222)


accuracy is equal to F1_micro for binary classification problem
       Iteration 1: 1.000000
       Iteration 2: 0.986301
       Iteration 3: 1.000000
       Iteration 4: 1.000000
       Iteration 5: 0.972222
Accuracy: 0.9917 (+/- 0.0222)


Precision = tp / (tp + fp). It measures "out of all *predictive positives*, how many␣
↪are correctly predicted?"
       Iteration 1: 1.000000
       Iteration 2: 1.000000
       Iteration 3: 1.000000
       Iteration 4: 1.000000
       Iteration 5: 1.000000
Precision: 1.0000 (+/- 0.0000)
```

(continues on next page)

(continued from previous page)

```
Recall = tp / (tp + fn). Recall (i.e. sensitivity) measures "out of all  *positives*,␣
↪how many are correctly predicted?"
       Iteration 1: 1.000000
       Iteration 2: 0.980769
       Iteration 3: 1.000000
       Iteration 4: 1.000000
       Iteration 5: 0.960784
Recall: 0.9883 (+/- 0.0313)
```

CHAPTER 9

SVM_predict.py

9.1 Description

Build SVM model from "train_file" and then predict cases in "data_file"

9.2 Options

--version show program's version number and exit

-h, --help show this help message and exit

-t TRAIN_FILE, --train_file=TRAIN_FILE Tab or space separated file (for tranining purpose, to build SVM model). The first column contains *sample IDs*; the second column contains *sample labels* in integer (must be 0 or 1); the third column contains *sample label names* (string, must be consistent with column-2). The remaining columns contain featuers used to build SVM model.

-d DATA_FILE, --data_file=DATA_FILE Tab or space separated file (new data to predict the label). The first column contains *sample IDs*; the second column contains *sample labels* in integer (must be 0 or 1); the third column contains *sample label names* (string, must be consistent with column-2). The remaining columns contain featuers used to build SVM model.

-C C_VALUE, --cvalue=C_VALUE C value. default=1.0

-k S_KERNEL, --kernel=S_KERNEL Specifies the kernel type to be used in the algorithm. It must be one of 'linear', 'poly', 'rbf', 'sigmoid', 'precomputed' or a callable. If none is given, 'rbf' will be used. default=linear

9.3 Input files format

TRAIN_FILE and DATA_FILE use the same format as below. the 2nd and 3rd columns in DATA_FILE can be consideres as **Original Label** and **Original Name**.

ID	Label	Label_name	feature_1	feature_2	feature_3	...	feature_n
sample_1	1	WT	1560	795	0.9716	...	feature_n
sample_2	1	WT	784	219	0.4087	...	feature_n
sample_3	1	WT	2661	2268	1.1691	...	feature_n
sample_4	0	Mut	643	198	0.5458	...	feature_n
sample_5	0	Mut	534	87	1.0545	...	feature_n
sample_6	0	Mut	332	75	0.5115	...	feature_n

9.4 Command

```
$ python3  SVM_predict.py -t lung_CES_5features.tsv  -d lung_CES_data_to_predict.tsv -
↪C 10
```

9.5 Output to screen

```
TCGA_ID         Ori_Label       Ori_name        Predict_Label   Predict_Name
TCGA-05-4244    unknown TP53_WT 1       Truncating
TCGA-05-4249    unknown TP53_WT 1       Truncating
TCGA-05-4250    unknown TP53_WT 1       Truncating
TCGA-05-4389    unknown TP53_WT 1       Truncating
TCGA-05-4390    unknown TP53_WT 1       Truncating
TCGA-05-4403    unknown TP53_WT 1       Truncating
TCGA-38-7271    unknown TP53_WT 1       Truncating
TCGA-38-A44F    unknown TP53_WT 0       Normal
TCGA-39-5030    unknown TP53_WT 1       Truncating
```

CHAPTER 10

gComposite.py

10.1 Description

This program Calculates these Composite Expression Scores. Compared to expression score of a single gene, *composite expression score* measure the overall activity of **a set of genes**. It is often used to measure the activity of a pathway or transcription factor.

It calculates these scores:

- Gene Set Variation Analysis (GSVA).[1]
- Single Sample GSEA (ssGSEA).[2]
- zscore[3]
- plage[4]

Note: The R package GSVA will be automatically installed and used to calculate these scores.

10.2 Options

--version	show program's version number and exit
-h, --help	show this help message and exit

[1] Hänzelmann S, Castelo R, Guinney J. GSVA: gene set variation analysis for microarray and RNA-seq data. BMC Bioinformatics. 2013;14:7. Published 2013 Jan 16. doi:10.1186/1471-2105-14-7

[2] Barbie DA, Tamayo P, Boehm JS, et al. Systematic RNA interference reveals that oncogenic KRAS-driven cancers require TBK1. Nature. 2009;462(7269):108-112. doi:10.1038/nature08460

[3] Lee E, Chuang HY, Kim JW, Ideker T, Lee D. Inferring pathway activity toward precise disease classification. PLoS Comput Biol. 2008;4(11):e1000217. doi:10.1371/journal.pcbi.1000217

[4] Tomfohr J, Lu J, Kepler TB. Pathway level analysis of gene expression using singular value decomposition. BMC Bioinformatics. 2005;6:225. Published 2005 Sep 12. doi:10.1186/1471-2105-6-225

-e EXPR_FILE, --expr_matrix=EXPR_FILE Tab-separated data matrix file containing gene expression values. The 1st row containing sample/patient IDs and the 1st column containing gene symbols(mut be unique). File can be compressed (.gz, .Z, .z, .bz, .bz2, bzip2).

-g GENE_FILE, --gene=GENE_FILE GMT file. The GMT file format is a tab delimited file format that describes gene sets (Each gene set is described by a name, a description, and the genes in the gene set). In the GMT format, each row represents a gene set. The first column is get set name (must be unique). The second column is brief description (can be 'na').

-k GROUP_FILE, --group=GROUP_FILE Group file (in CSV format). First column is sample ID, second column is group ID

-s SAMPLE_FILE, --sample=SAMPLE_FILE Sample list file containing sample IDs. Each row can be a single sample ID, a comma-separated sample IDs or a space-separated sample IDs. Sample IDs must match exactly to those in the data matrix file. If omitted, calculated activity scores for *all* the samples. File can be compressed (.gz, .Z, .z, .bz, .bz2, bzip2). default=none (All samples will be used)

-l, --log If True, will do log2(x+1) transformation for gene experssion values. Must set to 'True' if expressin values are RNA-seq count. default=False

-p N_THREAD, --processor=N_THREAD Number of processors to use when doing the calculations in parallel. default=0 (use all available processors)

-o OUT_FILE, --output=OUT_FILE The prefix of the output file.

10.3 Input files (examples)

- Gene expression table. Example: lung_expr.81genes.tsv
- Gene list in GMT format. Example: lung_p53_target.gmt
- Group file. Example: lung_group.csv

10.4 Command

```
$ python3 gComposite.py -e lung_expr.81genes.tsv -g lung_p53_target.gmt -k lung_
↪group.csv -o lung
```

10.5 Output files

- output.R : R script to run GSVA package
- output.mat.tsv : Data that is actually used. Might be the same as the input "lung_expr.81genes.tsv", or just a subset of "lung_expr.81genes.tsv".
- output_combined.tsv : comma-separated composite expression score (group IDs were also included)
- output_gsva.csv : GSVA scores

- output_pca.csv : First two principal components of PCA.
- output_plage.csv : PLAGE scores
- output_ssgsea.csv : ssGSEA scores
- output_zscore.csv : Z-scores

Note: The file "output_combined.tsv" contains everything you need for SVM model building and testing.

10.6 References

CHAPTER 11

Version 1.0.0

- July 8, 2020 at 3:09:33 PM CDT
- Initial release