r-SVM 2.0 for Linux

A User Manual

By:

Benoit Valin, Eng.
Institute of Bioinformatics and Department of Automation
Tsinghua University, Beijing 100084, China
e-Mail: bvalin@tsinghua.edu.cn

Based on :

r-SVM 1.0 by:

Xuegong Zhang, Ph.D.
Institute of Bioinformatics and Department of Automation
Tsinghua University, Beijing 100084, China
e-Mail: zhangxg@tsinghua.edu.cn

and on :

SVMTorch II v 1.77 by:

Ronan Collobert
IDIAP
CP 592, rue du Simplon 4, 1920 Martigny, Switzerland
e-Mail: ronan.collober@idiap.ch

Important Notice:

1. Using r-SVM implies the agreement of the license terms. See the README.txt for details.

2. It is strongly recommended that users learn the basic principle of SVM before using it. A detailed description of r-SVM can be downloaded from our r-SVM website, and further study can follow the references provided thereby. Applying advanced methods (such as r-SVM) without really understanding its core idea might be risky and lead to false discoveries.

0. Table of Contents

1.0. r-SVM 2.0 model

The r-SVM 2.0 package is aimed to solve binary (two-class) classification problems. Although there are ways of doing continuous regression with SVM and also ways of doing multiple classification with an ensemble of SVMs, and the same SVM technique can be applied to the classification genes rather than samples, the current version hereby released is only for the purpose of binary classification of the samples according to their gene expression data. The purpose of this section is to detail the intended uses and functionalities of this package.

The r-SVM 2.0 package is broken down into five (5) executable files; figure 1 illustrates the interrelation of each module in this package.

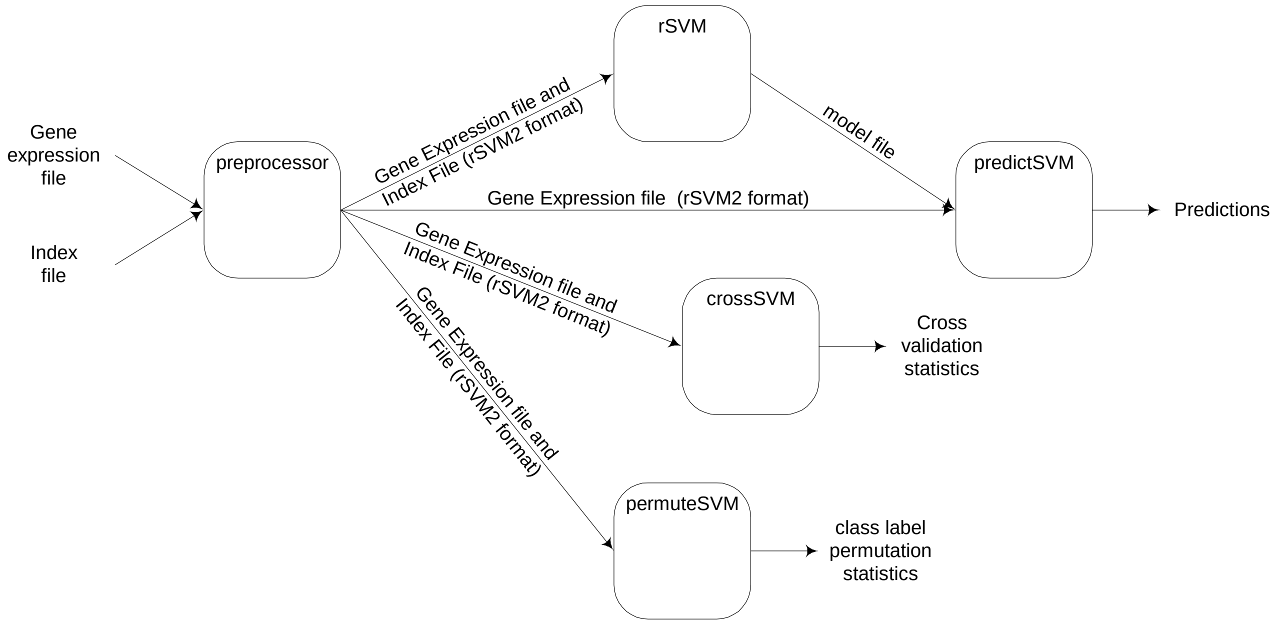

rSVM
Gene expression file
Index file
preprocessor
Gene Expression file and Index File (rSVM2 format)
model file
predictSVM
Gene Expression file (rSVM2 format)
Predictions
Gene Expression file and Index File (rSVM2 format)
crossSVM
Cross validation statistics
Gene Expression file and Index File (rSVM2 format)
permuteSVM
class label permutation statistics

Fig. 1: r-SVM 2.0 – Module interaction diagram

1.1. Preprocessing the data (preprocessor)

Preprocessing the micro-array data is a very important step for any further studies. We recommend using the filters offered by the r-SVM 2.0, dChip or Affymetrix's oligo micro-array data. For cDNA array data, we do not give specific recommendations, but users should take their effort to make sure that the data are properly preprocessed.

1.2. Building SVM Models (rSVM)

The simplest application of r-SVM 2.0 is to build a series of SVM classification models with different number of genes according to the selection by r-SVM. These models can be applied on later-on samples for prediction purpose or for testing purpose. However, by building models only, you cannot have a reliable assessment of the performance of the models.

1.3. Making prediction based on SVM Models (predictSVM)

Making classification predictions of new samples based on a verdict rendered by a r-SVM 2.0 model can be done by preprocessing the new data, with the proprocessor, to select and order the genes into the format the model was created, using rSVM, and then using predictSVM to obtain classification results.

1.4. Cross Validating the SVM Models (crossSVM)

If your samples do not allow you to get an independent test set, you can use cross validation to assess the accuracy of the SVM models, from which you can get the idea about how strong the classification signal is in the data set as revealed by linear SVM.

Note: Since cross validation generates a set of SVM models at each gene selection level, the cross validation error rate is not the estimate of accuracy of any specific model of them, but rather an overall estimate of how strong the signal is and how well SVM can work on this data set. If you are aiming at deriving one single SVM classification model for future use at a certain gene selection level, you need to build the SVM model as described in (1.2). You cannot have a direct estimate of the performance of this model, but you can have an idea about how well it could be according to the cross validation performance on the same data set.

1.5. Assessing the significance of the cross validation error rate (permuteSVM)

To assess the significance of the cross validation error rate, you can do permutation experiment to get an estimated permutation p-value. This is essential if your purpose is to discover the existence of predictive signals in the data set. This should be done prior to any effort aiming for building a practical prediction model.

2.0. Basic Conventions

Throughout this instruction, we'll try to follow the basic conventions of UNIX. For example, when describing the syntax of a command, the contents in "[ ]" in a command line is an optional parameter of the command, the underlined part of a command line option should be given by the user according to his/her need (e.g., a file name), and the other parts of the command line should be typed exactly as illustrated.

Since some of the computations might take a very long time (say, from hours to weeks depending on the type of work and the size of your data), This release of r-SVM provides a progress bar to indicate the state of the process.

2.1. Basic Terms

Gene

We use "gene" in a broad sense. It is simply the basic unit of the features to be investigated. For example, for data from Affymetrix type of microarrays, we take each probe-set as one gene; for array CGH data, we take one BAC as one gene, and so on.

SVM - Support Vector Machine

By default, we mean linear SVM when we say SVM, since for current micro-array data, it is not reasonable to use non-linear version of SVMs due to the high risk of overfitting.

Sample

The data of a sample means the data vector whose components are the expression values of all the genes at this sample measured by the micro-array experiment. A sample can be either labeled, meaning we know which class the sample belongs to, or unlabeled (or unknown), meaning that we don't know (or pretend not to know) the class label of this sample.

Training Data

The sample set in which all the samples are labeled, and will be used for designing (training) the classifier.

Test Data

The sample set in which all samples are unlabeled (the label information will not be used although it is there), and will be used to test the performance of a classifier according to whether the class label predicted by the classifier is the same with the known label.

Unknown Data

The sample set in which all the samples have not been assigned a class label. A classifier can be applied on the unknown data to make predictions. The difference with a test data set is that we cannot judge whether the prediction is correct or not.

2.2. Data Formats

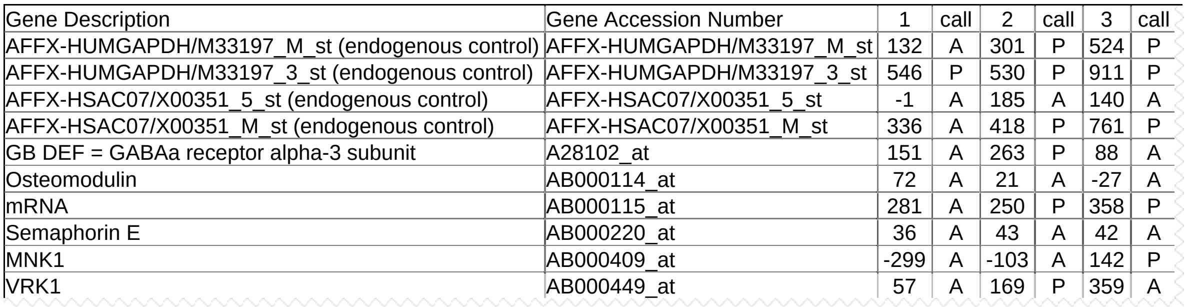
2.2.1. Gene Expression Data Files

All the gene expression data of all samples should be contained in a single ASCII file, with each row representing one gene, and each column (or two columns for some format, see below) representing one sample (one case). Each item in the data file is delimited with a tab ('\t'). The expression file can be the output of some popular micro-array data processing softwares such as dChip (Li & Wong).

The preprocessor accepts five (5) file formats; the paragraphs bellow illustrates each of the file formats.

2.2.1.1. The Whitehead Format (WI)

The old format used by Whitehead Institute for the leukemia (ALL/AML) data, probably generated with the GeneChip software by Affymetrix. Here is a sample:

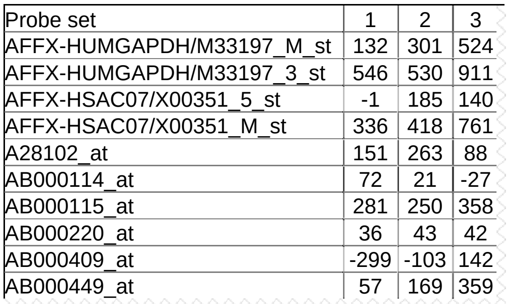


Gene Description	Gene Accession Number	1	call	2	call	3	call
AFFX-HUMGAPDH/M33197_M_st (endogenous control)	AFFX-HUMGAPDH/M33197_M_st	132	A	301	P	524	P
AFFX-HUMGAPDH/M33197_3_st (endogenous control)	AFFX-HUMGAPDH/M33197_3_st	546	P	530	P	911	P
AFFX-HSAC07/X00351_5_st (endogenous control)	AFFX-HSAC07/X00351_5_st	-1	A	185	A	140	A
AFFX-HSAC07/X00351_M_st (endogenous control)	AFFX-HSAC07/X00351_M_st	336	A	418	P	761	P
GB DEF = GABAa receptor alpha-3 subunit	A28102_at	151	A	263	P	88	A
Osteomodulin	AB000114_at	72	A	21	A	-27	A
mRNA	AB000115_at	281	A	250	P	358	P
Semaphorin E	AB000220_at	36	A	43	A	42	A
MNK1	AB000409_at	-299	A	-103	A	142	P
VRK1	AB000449_at	57	A	169	P	359	A

The first line contains the name (tag) of the samples. For each sample, there are two columns: the first column contains the expression values of the genes, and the second column contains either a 'A' or 'P' for each gene, indicating whether the gene is recognized as "Absent" or "Present" by the Affymetrix software. The first and 2nd columns of the file are the gene description and the probe set tag.

2.2.1.2. Cheng Li's compact format (CHENG)

The simplest expression output format provided by dChip. Here is a sample:

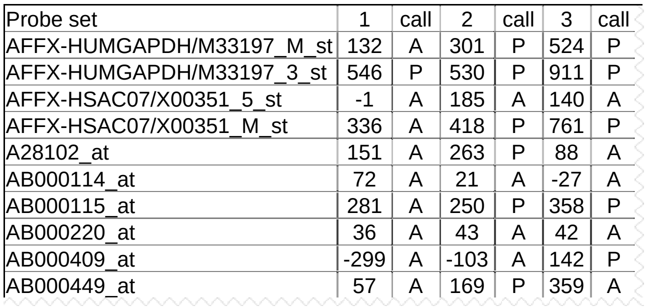


Probe set	1	2	3
AFFX-HUMGAPDH/M33197_M_st	132	301	524
AFFX-HUMGAPDH/M33197_3_st	546	530	911
AFFX-HSAC07/X00351_5_st	-1	185	140
AFFX-HSAC07/X00351_M_st	336	418	761
A28102_at	151	263	88
AB000114_at	72	21	-27
AB000115_at	281	250	358
AB000220_at	36	43	42
AB000409_at	-299	-103	142
AB000449_at	57	169	359

It is similar to the WI format except that only one column is used for containing the probe set tags. Also there is only one column per sample, containing only the expression values (without any 'A' or 'P' information).

2.2.1.3. Cheng Li's complete format (CH2)

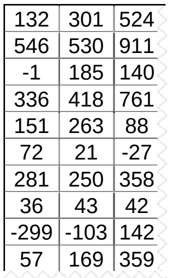
Similar with CHENG format, except that the Absent/Present information is available in the file so that each sample has two columns. Here is a sample:



Probe set	1	call	2	call	3	call
AFFX-HUMGAPDH/M33197_M_st	132	A	301	P	524	P
AFFX-HUMGAPDH/M33197_3_st	546	P	530	P	911	P
AFFX-HSAC07/X00351_5_st	-1	A	185	A	140	A
AFFX-HSAC07/X00351_M_st	336	A	418	P	761	P
A28102_at	151	A	263	P	88	A
AB000114_at	72	A	21	A	-27	A
AB000115_at	281	A	250	P	358	P
AB000220_at	36	A	43	A	42	A
AB000409_at	-299	A	-103	A	142	P
AB000449_at	57	A	169	P	359	A

2.2.1.4. Pure Data Format (PURE)

This is the simplest format, which is virtually just a matrix, with each row representing one gene and each column representing one sample. No sample tag or probe set (gene) tag information is kept in the file. Here is a sample:



132	301	524
546	530	911
-1	185	140
336	418	761
151	263	88
72	21	-27
281	250	358
36	43	42
-299	-103	142
57	169	359

Since this data format does not provide any information concerning the gene and sample names, the preprocessor automatically generates sample and gene names in the following format:

Type	Index	Generated name
Gene	X	Gene_X
Sample	Y	Sample_Y

2.2.1.5. r-SVM 2.0 Data Format (RSVM2)

In order to simplify the design of each module within the r-SVM 2.0 package, the preprocessor converts the above mentioned data formats into a common data format. A slightly modified version (without the “Probe set” tag in position (0;0)) of the CHENG has been chosen to be the common file format.

2.2.1.6. A note about the Affymatrix chip

The Affymatrix chips usually contain probe sets named as "AFFX-***" which are probe sets for quality-controlling purpose only. The preprocessor removes these genes during the conversion process.

2.2.2. Class Index Files

Since no class labeling information is kept in the expression data file, the class labeling information can be inputted to the preprocessor.

2.2.2.1. Index files

This formats uses two (2) files (one for each class) to indicate which sample is of which class. An index file contains the relative indexes of the samples that belong to the same class. Please note that the index begins from 0; and that each index should be on a separate line.

2.2.2.2. Class file

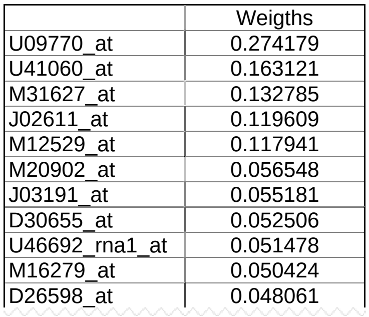
This format uses only one (1) file containing a combination of “1” or “-1” to indicate which sample is of which class. Please note that theres should be the same amount of values than the number of samples in the gene expression file; and that each index should be on a separate line.

2.2.3. SVM Model Files

Files named as "*.model" are SVM model files obtained by training SVM with some training samples. These model files can be used for test/prediction purpose. They are binary files so that you cannot directly read or edit them. See 4 and 5 for how to use these files.

2.2.4. SVM Gene Sorting Files

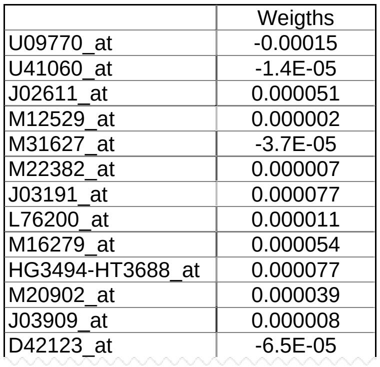
Files named as "*.sort" are SVM gene sorting files. File "1000g.sort" means the sorting of the 1000 genes used by SVM. This is an ASCII text file, in which the content look likes the following example:



	Weigths
U09770_at	0.274179
U41060_at	0.163121
M31627_at	0.132785
J02611_at	0.119609
M12529_at	0.117941
M20902_at	0.056548
J03191_at	0.055181
D30655_at	0.052506
U46692_rna1_at	0.051478
M16279_at	0.050424
D26598_at	0.048061

2.2.5. SVM Gene Selection Files

Files named as "*.genes" contain the gene (probe set) names (tags) of genes being selected to generate a SVM model file accompanied by their relative weight within the SVM model. This is an ASCII text file, in which the content look likes the following example:



	Weigths
U09770_at	-0.00015
U41060_at	-1.4E-05
J02611_at	0.000051
M12529_at	0.000002
M31627_at	-3.7E-05
M22382_at	0.000007
J03191_at	0.000077
L76200_at	0.000011
M16279_at	0.000054
HG3494-HT3688_at	0.000077
M20902_at	0.000039
J03909_at	0.000008
D42123_at	-6.5E-05

Since SVM models are feature and feature order sensitive. Each time a model file is generated, a gene selection file is also generated. The gene selection files can then be used as an input to the preprocessor to process new gene expression files into gene expression files compatible with the generated model (generated with fewer genes and in different order).

Users will be interested in the gene selection files only if they want to investige the gene selected by r-SVM 2.0.

2.3. About Gene Selection

r-SVM 2.0 offers different schemes to control gene selection at each level of recursion. These features are details in later sections of this document. If you urgently want to do selection schemes that are not currently offered, please contact the author for a possible especially personalized version of r-SVM 2.0 for you!

3. Using the r-SVM 2.0 package

3.0. Preprocessing your data

3.0.1. The command line synopsis

```
Syntax:
      preprocessor InputFileName DataFormat OutputFileName
                   [CLASS-INFO] [FILTERS...]
Base Parameters:
      InputFileName  :  The gene expression file.
      DataFormat     :  The data format of the gene expression file.
                        Values are:
                           WI    - Whitehead format
                           CHENG - Cheng Li's compact format
                           CH2   - Cheng Li's complete format
                           PURE  - Pure format
                           RSVM2 - r-SVM 2.0 format
      OutputFileName :  The output r-SVM 2.0 gene expression file
Class Information:
      -idx1 classFile: The index file Containing the class information of
                       each sample (either 1 or -1).
      -idx2 index1File index2File: Where each index File contains the
                                   indexes of the samples in each class
                                   (starting at index 0).
      -idxn name1File name2File  : Where each index File contains the names
                                   of the samples in each class
          ****NOTE: The class information will be remapped and saved under
                    the following name (to prevent any confusion with any
                    other index file): OutputFileName.idx
Gene selection Information:
      -genes GeneSelectionFileName : The gene selection file (.genes) used
                                     to Prepare a new gene expression file
                                     to be used with a created model.
                                     Outputted by SVM training (rSVM).
Filters:
      -topcut value : Excludes the top "value" genes (before other
                      filtration).
      -smplexc name : Excludes the sample named "name".
      -geneexc name : Excludes the gene named "name".
      -smplthresh min max : Excludes the samples that do not respect
                            the following rule:
                                min <= gene value <= max
                    NOTE:     (max - min) != 0
      -genethresh min max : Excludes the genes that do not respect the
                            following rule:
                                min <= sample value <= max
                    NOTE:     (max - min) != 0
      -nfold n     : Excludes the genes that has a sample that do
                     not respect the following rule:
                          n < (max(sample value) / min(sample value))
      -std mean stddiv : Standardize all the gene expression to a mean of
                         "mean" and a standard deviation of "stddiv".
                     NOTE:          stddiv > 0
      -topsel value : Keep only the "value" top genes (after other
                      filtration).
```

This program will output entry files necessary for the use of all the other programs in the r-SVM 2.0 package.

3.1. Doing Recursive Gene Selection and Classification

Doing Recursive Gene Selection and Classification with r-SVM 2.0 can be done through the **rSVM** program. This program creates a series of SVM gene-selection lists, and SVM classification models built on these genes, without any procedure to evaluate the performance of the built models.

3.1.1. Gene selection algorithms

This new version of r-SVM provides four (4) gene selection algorithms. The algorithm can be applied to each of the SVM learning programs (rSVM, crossSVM and permuteSVM); with exception.

3.1.1.1. Based on weight

$$UserSpecified \le \frac{WeigthOfSelectedGenes}{WeigthOfAllTheGenes}$$

Where

$$0 < UserSpecified < 1$$

NOTE: Since this algorithm does not guarantee the same gene selection count at each level, this gene selection algorithm can only be used in rSVM and not in crossSVM or permuteSVM.

3.1.1.2. User specified

$$Selection := [\, UserSel_k,\ k = 0, 1 \,..\, n \,]$$

NOTE: The user must provide a file with the gene selection counts in it.

3.1.1.3. Fixed amount (Default)

$$Selection := [ALL, 1000, 500, 200, 100, 50, 30, 20, 10, 5]$$

3.1.1.4. Divide by half

$$Selection_{k+1} = \frac{1}{2} Selection_k$$

Where

$$Selection_0 = ALL$$

3.1.2. The command line synopsis

```
Syntax  :
       rSVM InputFileName IndexFile [ ...SELECTION ALGORITHIM... ]

Base Parameters :
       InputFileName  : The gene expression file (in rsmv2 format).
       IndexFile      : Contains belonging class of each sample
                        (either 1 or -1).

Selection Algorithm:
      The options in this program relate directly to the recursive gene
      selection algorithm. The recursively will stop when less than 2
      genes are selected or that the number of genes does in a new
      selection does not decrease the gene count. There are 4 types of
      gene selection algorithms:

      -ratio WeigthRatio : This algorithm will select genes until the
                           equation verified:

                               SumOfWeigth(selected genes)
                WeigthRatio <= ---------------------------
                                 SumOfWeigth(all genes)

                NOTE: Progress cannot be established for this method.


      -user SelectionFile : This algorithm will recursively select the
                            number of genes specified in the
                            'SelectionFile'. The file should contain 1
                            value per line and be in decreasing order.

      -fixed  :  This will select a fixed amount of genes at each
                 recursion. The values are:
                      [ALL; 1000; 500; 200; 100; 50; 30; 20; 10; 5]
               NOTE:  This algorithm is equivalent to r-SVM v.1.0
                      recursive gene selection algorithm.

      -divbyn  n  :  This algorithm will select 1/n genes at each new
                     recursion.
                          i.e. Selection(t+1) = Selection(t) / n
```

This program will output a series of SVM model files and gene selection files corresponding to each generated model.

3.2. Testing and Predicting New Samples with Built SVM Models

3.2.1. Testing Built SVM Models

Once you have an r-SVM 2.0 classification model, you can use an independent data set to test the accuracy of this model. The test set should be drawn independently from the same population as the training samples, and should have exactly the same file format and gene order as the training data file (this can be achieved by inputting the gene selection file during the preprocessing of your data).

3.2.2. Predicting New Samples with Built SVM Models

Predicting new samples with a built SVM model is exactly the same as the testing procedures, except that you don't know the correct class labels for the testing samples.

3.2.3. The command line synopsis

```
Syntax:
      predictSVM ModelFile IndexFile ResultFile

Base Parameters:
      ModelFile     :  The generated model file created by rSVM
      InputFile     :  The gene expression file (in rsmv2 format) to predict.
      ResultFile    :  The output prediction result file
```

This program will output a result file containing, on each line, the name of the sample followed by its predicted class value.

3.3. Doing Cross-Validation

It is important to note that in order to investigate the underlying classification relationship between the raw data and the classes being investigated, the steps for gene selection (if any) should be taken, as a part of the whole classification system, which means the gene selection steps should also be cross validated. Thus, the whole procedure of r-SVM cross validation can be broken down into three (3) steps, as follows:

1. Leave one example (sample) out from the sample set
2. Do a r-SVM gene selection and classification on the new sample set
3. Test the obtained series of models on the left-out example

Every example is left out once. The cross validation error rate is estimated by counting the proportion of the examples being mistaken in the test when it is the left-out example. See the reference for details. The implementation of the cross validation procedure is rather straightforward though, by executing the following command:

NOTE: The current version (2.0) of r-SVM allows only leave-one-out cross validation.

3.3.1. The command line synopsis

```
Syntax  :
       crossSVM InputFile IndexFile StatFile [ ...SELECTION ALGORITHM... ]

Base Parameters :
      InputFile      :  The gene expression file.
      IndexFile      :  The index file Containing the class information of
                        each sample (either 1 or -1).
      StatFile       :  Output file name containing the statistics of the
                        crossSVM validation.

Selection Algorithm:
     The options in this program relate directly to the recursive gene
     selection algorithm. The recursively will stop when less than 5 genes
     are selected or that the number of genes does in a new selection does
     not decrease the gene count. There are 3 types of gene selection
     algorithms:

     -user  SelectionFile  :  This algorithm will recursively select the
                              number of genes specified in the
                              'SelectionFile'. The file should contain 1
                              value per line and be in decreasing order.

     -fixed : This will select a fixed amount of genes at each recursion.
              The values are:
                  [ALL; 1000; 500; 200; 100; 50; 30; 20; 10; 5]

              NOTE: This algorithm is equivalent to r-SVM v.1.0 recursive
                    gene selection algorithm.

     -divbyn n :  This algorithm will select 1/n genes at each new
                  recursion.
                       i.e. Selection(t+1) = Selection(t) / n
```

This program will output results into two distinct formats.

1. The results of each leave out case will be saved in a folder named "Leave_SAMPLENAME". In this folder the gene sort file and an error file (errors.note) will be saved. In this error file, each recursion case will be quoted along with classification results and SVM outputs.
2. The statistic file contains the average error rates (over all, positive and negative) at each recursion level.

3.4. Doing Permutation Experiments to Assess the Significance of R-SVM Results

If the classification signal in your data is not so strong, and especially when you sample size is not very large, you might need to do permutation experiments to get a strict estimation of how significant the classification you've got is. This is very important for serious scientific studies. See the references for more details.

Please note that each permutation test take approximately the same amount of time as the cross validation procedure. So if you want to do 100 runs of random permutation, it will cost you 100 times more time. If you have a cluster, you might want to distribute the permutation experiments on several clusters. The easiest way to do this is to execute a job of permuteSVM on each node of your cluster, but remember to use non-overlapping selection of random seeds for these jobs.

3.4.1. The command line synopsis

```
Syntax  :
       permuteSVM InputFile IndexFile StatFile NbPerms
                   [ ...SELECTION ALGORITHIM... ]  [ ...OPTION... ]

Base Parameters :
       InputFile      :  The gene expression file (in rsmv2 format).
       IndexFile      :  The index file Containing the class information of
                         each sample (either 1 or -1).
       StatFile       :  Output file name containing the statistics of the
                         permutation experiments.
       NbPerms        :  The number of permutations to execute

Selection Algorithm:
       The options in this program relate directly to the recursive gene
       selection algorithm. The recursively will stop when less than 5
       genes are selected or that the number of genes does in a new
       selection does not decrease the gene count. There are 4 types of
       gene selection algorithms:

       -user  SelectionFile  :  This algorithm will recursively select the
                                number of genes specified in the
                                'SelectionFile'. The file should contain 1
                                value per line and be in decreasing order.

       -fixed : This will select a fixed amount of genes at each recursion.
                The values are:
                   [ALL; 1000; 500; 200; 100; 50; 30; 20; 10; 5]

                NOTE: This algorithm is equivalent to r-SVM v.1.0 recursive
                gene selection algorithm.

       -divbyn n :  This algorithm will select 1/n genes at each new
                    recursion.
                           i.e. Selection(t+1) = Selection(t) / n
Option:
       Random Control:
              -seed s: Random number generator seed ‘s’. If none
                       specified, current time will be used.
```

3.5. Sample Script and Sample Data

This version r-SVM is distributed with a sample script and a sample dataset. In this section we will detail each sample file.

WARNING:
The data and results are meaningless; the files are provided to illustrate the functionalities of the programs, AND NOTHING MORE!

File Name	Description
demodata_ch2.xls	Sample Data set
Class_a.idx	Indexes of the “class A” samples
Class_b.idx	Indexes of the “class B” samples
Class_a+Class_b.idx	Class labels (1 or -1) of all the samples
Class_Names1.idx	Names of the “class A” samples
Class_Names2.idx	Names of the “class B” samples
SampleScript.sh	A pilot script to run the demo.

4. Notes from the author

4.0. Running r-SVM.2 in quiet mode

Since some processes might take hours, days or weeks, it is possible to launch the process with the help of “**nohup**” which will start the process and not permit it to be stopped by user logoff. Hence if you wish to launch a process on your main frame computer (at home) from you laptop while you’re driving down I-5 at 145 miles an hour in your beige 1973 Volvo 1800ES, you can use “**nohup** command_to_run” to make sure that the process will not be stopped when your laptop flies out the car’s window and gets pulverized on the curb.

4.1. User Display

In this new version of r-SVM, the run-time screen print out of information was stripped. A progress bar is now being displayed to give a user status update. In the case that the process is launched in quiet mode, a “progress.txt” file is created to indicate the process state.

4.2. Command line parameters

In this new version of r-SVM, the programs were stripped of the barrage of questions that was characteristic to the previous version and replace them with command line parameters. The purpose of this change was to enable the user to write bash shell scripts to orchestrate or automate the usage of the multiple executables.

4.3. Final thought

The author is 150 % **NON-RESPONSIBLE** in case of data loss or data corruption that could result in the miss use of a computer, software or firearms. The quote of the day is: “Back-up your stuff or Pack-up your stuff”.