PML manual

Runyu Jing, Jing Sun, Yuelong Wang, Menglong Li

August 7, 2013

Contents

PML manual 1
1. Introduction 2
2. License 2
3. Installation 2
3.1 Requirement 2
3.2 Install PML-Desktop 3
3.3 Install PML-Server 3
3.3.1 Install BOINC-Server 3
3.3.2 Integrate PML into boinc-server 6
3.3.3 Install boinc-client 6
4. The use of PML 7
4.1 Quick start 7
4.2 Write script 7
4.3 Add machine learning methods to PML 12
4.3.1 Add WEKA Method 12
4.3.2 Integrate 3rd-party program 12
4.4 Parameter optimization 16
4.4.1 Parameter optimization for weka 16
4.4.2 Parameter optimization for 3rd-party program 17
4.5 Manage the machine learning methods 18
4.6 Use the independent test dataset 19
4.7 Specify data for cross validation 19
4.8 Outputs 19
4.9 Examples 20
4.10 Recover and Reset an Experiment 21
5. Some operating suggests 22
5.1 Make input script more readable 22
5.2 Control the process of PML 22
5.3 Configure PML-Server 23
5.4 Manage Clients Remotely 24
5.5 Packing the tasks 25
6. Development and testing environment 25
7. Conclusion 26

1. Introduction

PML is a toolbox written in Perl, and it can use programs (like WEKA, Waffles, etc.) as methods to achieve dimensionality reduction, modeling, parameter optimization, and cross validation in parallel for data. Users only need to write a simple text as input, and PML will analyze the input and proceed the tasks automatically. Currently PML has two sub-versions: PML-Desktop and PML-Server. PML-Desktop could run on a single machine in parallel, and PML-Server could use the BOINC platform that enables parallel computing among machines running on different operating systems (Windows/Linux).

2. License

3. Installation

PML is designed to reduce the use of other modules, thus PML-Desktop can be installed simply. However, PML-Server needs the BOINC-Server, thus some components need the installation and initialization. In this chapter, the installations for the two sub-versions are offered as follows:

3.1 Requirement

PML uses these modules of Perl:

File::Copy	Cwd
threads	threads::shared
Scalar::Util	Archive::Tar

These modules are included as part of Perl in the version 5.16.3.1 and later ones. If the Perl in users' computer is an old version, the modules need to be installed manually (the details of the installation could be found in subsection Install PML-Desktop).

Besides, PML also needs the JAVA JRE (http://java.com/en/download/index.jsp) for WEKA.

3.2 Install PML-Desktop

In Windows, no more installations will be needed if the latest version of Perl interpreter (such as Active Perl or Strawberry Perl) has been installed. Moreover, most of the Linux systems have integrated Perl, thus users only need to check the installation of modules. There are two simple ways to check the installation of the modules:

1. Input **perl -e "use MODULENAME;"** in command line window. The **MODULENAME** is the name of the module to be tested (simply input: perl –e "use Cwd;" if the target module is Cwd), if no warning occurs, that means the module has been installed.
2. An alternative way is that input **perldoc MODULENAME**, if the help text of the module occurs, that means the successful installation of the module.

Missing modules can be installed in the following ways:

1. If Active Perl is utilized, users can use the PPM (Perl Package Manager) to install the modules
2. In other conditions, open a command line window and type in
   cpan **MODULENAME**
   or type in
   perl -MCPAN -e shell
   install **MODULENAME**

Moreover, users need to install the JAVA JRE (http://java.com/en/download/index.jsp) for WEKA.

3.3 Install PML-Server

PML-Server uses BOINC to achieve grid computing, thus the installation of BOINC-Server is necessary. Since the BOINC-Servers need Linux platform, at least one machine running on Linux is required. PML-Server could be installed by the following steps:

3.3.1 Install BOINC-Server

Two ways for the installation are provided.

1. Users could install the BOINC-Server by the official instruction:
http://boinc.berkeley.edu/trac/wiki/ServerIntro

2. Alternatively, users can follow the procedure provided below.

Prepare installation of boinc-server from source code

Some software and libraries need to be installed for Boinc-server. Users can find the requirement from http://boinc.berkeley.edu/trac/wiki/SoftwarePrereqsUnix or follow the suggestions:

Ubuntu/Debian

In Ubuntu/Debian, the apt-get command would help users to install software from APT (Advanced Packageing Tool) source on internet. List as below:

subversion	make	m4	pkg-config
dh-autoreconf	libssl-dev	python-mysqldb	php5-mysql
php5-gd	php5-dev	libmysqld-dev	mysql-server
tcsh	libfcgi-dev	libnotify-dev	libxss-dev
libxmu-dev	libxi-dev	libgtk2.0-dev	libsqlite3-dev
libapache2-mod-php5	libglut3-dev/freeglut3-dev	libjpeg-dev/ libjpeg8-dev	

Note that some libraries support the same functions for BOINC-Server, such as the libglut and freeglut, thus only one of them is needed. Besides, the versions of some software might be updated with the update of the APT source, thus the names in the list above might be changed. For example, the libglut3 might possibly be changed to libglut4 in the future.

CentOS/Fedora

In CentOS/Fedora, most software/libraries can be installed by the Yellowdog Package Manager (yum):

Subversion	gcc	gcc-c++	make
Autoconf	automake	freeglut-devel	libnotify-devel
m4	libtool	libXi-devel	libstdc++static
MySQL-python	php-mysql	php-gd	php-cli
openssl-devel	mysql-devel	mysql-server	httpd
php	libXmu-devel	fcgi-devel	libxss-devel
libjpeg-turbo-devel	libXi-devel		

Note that some of the libraries might be absent in the yum source. Users could get the software/libraries by the links below and install them manually:
fcgi-devel : http://www.fastcgi.com/drupal/node/5
libjpeg-devel : http://www.ijg.org
libxss-devel : http://cygwin.mirrorcatalogs.com/release/X.Org/libXScrnSaver/libXss-devel/

Install boinc-server

After the installation of the required software/libraries, users could use this command line to get

the source code of boinc:

svn co http://boinc.berkeley.edu/svn/trunk/boinc

After that, a folder named 'boinc' will be created and system will download the source into the folder automatically. Then type

cd boinc

And begin to install:

./auto_setup

./configure -C --disable-client

make

make install

Notes:

1. Last step (**make install**) might need the root permission if default installation path is used.
2. The first step, downloading the source code, needs the software sub-version installed first.
3. If some modules is missing, related warnings will be prompted when execute configure command. This trick might be useful to catch them:
   ./configure -C --disable-client 1>install_log.txt 2>err_log.txt
   If no warning or error occured, the file err_log.txt would be empty.
4. In the process of configuring, if the system shows that some libraries, which have been installed, cannot be found, try the command **updatedb** under the root permission and delete the file config.cache under the boinc root path before rerunning the configuring.

Run make_project script

After install the base of the BOINC-Server, the script **make_project** in the subfolder 'tools' is available to create a project which can generate, sent tasks and receive results. In general, the **make_project** script works as follow:

<boinc-server_path>/tools/make_project --url_base project_url pml 'PML@grid'

Where the **project_url** means the url of the project, and the Clients could link to the Server by it. The url need to be a statics address (like http://1.2.3.4) if no DNS service has been set (this situation is general in local area network). The **pml** and '**PML@grid**' is the short name and full name of the project.

Then, open the project folder (default is ~/projects/pml), do initializations follow the details in the file 'pml.readme' in the project folder.

Notes:

1. The **make_project** script would create a database in MySQL. Users can specify the database user, host and password (if needed) by the options provided by **make_project**:
   --db_name --db_user --db_passwd --db_host
   Thus, the MySQL server can be installed in other machine for some requirement.
2. If errors occur and users want to retry the script, the two options need to be added:
   --delete_prev_inst --drop_db_first
3. If the Linux OS is CentOS/Fedora, the project might be inaccessible from Internet browser

because the set of Firewall and SELinux, users could close them if only for checking.

4. If 403 error occurred when access into the project by browser, that might due to the permission of group. Consequently, make sure that apache (CentOS/Fedora) or www-data (Ubuntu/Debian) is in the group where the project was located and the permission of all the parent folders is right.
5. A 3rd-party package in the APT source, Boinc-server-maker, is provided for the installation of Boinc-server. However, this package may cause some bugs when it was tested (some files absent such as **antique_file_deleter** of version 7.0.65+dsfg-3) and the version is relatively old, thus the installation from source code is recommended.

3.3.2 Integrate PML into boinc-server

After the initialization of projects, the components of PML-server can be integrated into the project easily by executing the file 'PML2BOINC.pl':
perl PML2BOINC.pl
Some prompts will be thrown out and users could finish the integration. Then, make the work path under the root of created project and run:
bin/xadd
bin/update_versions
bin/start

3.3.3 Install boinc-client

The BOINC-Client is needed to be installed on the Client machines that the tasks will be executed on. The packages of the BOINC-Client and BOINC-Manager (the GUI of boinc client) could be found and installed easily by using the yum or apt-get except some Linux OS with older kernel. For example, the BOINC-Client cannot be found in yum source for CentOS (version 6.4), and users need to install the BOINC-Client manually in this situation.

Users can follow the steps when install the BOINC-Server and ignore the software/libraries correspond to php/mysql/apache/httpd/ssl. Then, change the option of the configuring command:
./configure -C --disable-server
Finally, the BOINC-Client will be installed after the command '**make install**'.

For the Client machine running on Windows, BOINC-Client is available by the official link:
http://boinc.berkeley.edu/download.php

After the installation, users could add the project by using the BOINC-Manager or BOINC-Client, then the installation and initialization are completed.

Notes:
If the number of Clients is too large, users can package the initialized BOINC-Client for other

machines. Use the package may reduce the time in installation and adding project, but the operating system of the Client machines need to be the same. More details could be found from http://boinc.berkeley.edu/wiki/Deploying_BOINC_on_networks

4. The use of PML

PML can be used easily. As a start, the 'Quick start' is recommended. Then the other functions could be understood accordingly.

4.1 Quick start

In this section, the 'Quick start' with PML-Desktop will be used in the following example. PML-Server also shares the same process as it uses the similar script as PML-Desktop does.

1. Make sure Perl together with its modules and JAVA JRE are installed.
2. Download and uncompressing pml.zip (or pml.tar.gz) into any empty folder. The related commands in Linux are:
   unzip pml.zip
   or
   tar -zxvf pml.tar.gz
3. Open a command line window and locate the work path under the folder (use cd command), then type in:
   perl pml_desktop.pl quick_start
4. The process would be completed in few seconds, and the results can be found in '<pml_root>/pml/results/quick_start/results.html'

Since the 'quick_start' script contains only a part of parameters, another script 'quick_start_full' is provided in the same path. This script contained most of the parameters that commonly used. Users could try it by changing the command line in step 3 to:

perl pml_desktop.pl quick_start_full

Note that running the script 'quick_start_full' needs more time.

More examples can be found in the folder 'pml/examples' and a README file is provided for the content of each example.

4.2 Write script

The scripts for PML are similar to the script of Quick start, and more examples are provided in the subchapter Examples.

All the parameters are listed below. Note that the **Example** in the table means an example value for the parameter. For example, the **Example** './data/iris.arff' of the parameter **FILE** could be used in a script as this format:

FILE=./data/iris.arff

Parameter Name	Description	Example	Necessity
	Comment		
NAME	The name of experiment	Try_pml	Yes
	Names of two experiments can not be the same.		
FILE	The path of input dataset	./data/iris.arff	Yes
STEP	Specify the steps of the processes.	clu,fea,tt	Yes
	The 'clu' means data cluster, and the 'fea' means variable selection. The sequence of the 'clu' and 'fea' could change. The 'tt' (training and testing) should be the last step		
CLUSTER_ARG	Cluster algorisms	1,2,3	Depend on STEP
	Should be corresponding with STEP. The sequence number here are corresponded with the contents in the file **<pml_root>/config/cluster.** More details are in the subsection Manage the machine learning methods		
CLUSTER_OUT_INSTANCES	The number of instances of the dataset after cluster	0.5,200	Depend on CLUSTER_ARG
	If the value is less than 1, the instances will be reduced in proportion. If one of the values is 'all', the step of cluster for this value will be skipped.		
CLUSTER_ARG_OPT	Parameter optimization of cluster	Introduce in the later	Depend on CLUSTER_ARG
	More details are in the subsection Parameter Optimization		
FEATURE_SELECT_ARG	Variable (feature) selection algorism.	1,2,3	Depend on STEP
	Should be corresponding with STEP. The sequence number here are corresponded with the contents in the file **<pml_root>/config/feature.** More details are in the subsection Manage the machine learning methods		
FEATURE_OUT_FEATURES	The number of the rest variables after variable selection	5,7	Depend on FEATURE_SELECT_ARG
	This parameter only affects methods which can rank variables. If one of the values is 'all', the step of variable selection for this value will be skipped.		
FEATURE_ARG_OPT	Parameter optimization of variable selection	Introduce in the later	Depend on FEATURE_SELECT_ARG
	More details are in the subsection Parameter Optimization		
TT_ARG	Modeling (train/test) algorisms	1,2,3	Yes
	Should be corresponding with STEP. The sequence number here are corresponded with the contents in the file		

	<pml_root>/config/traintest. More details are in the subsection Manage the machine learning methods		
TT_ARG_OPT	Parameter optimization of modeling	Introduce in the later	Depend on TT_ARG
	More details are in the subsection Parameter Optimization		
OUTER_FOLDS	Outer n-fold	5	Yes
	The test data is needed if the value is 1.		
INNER_FOLDS	Inner n-fold	5	No
	The valued should be no less than 2.		
INDEPENDENT_DATA	The path of independent test dataset	./data/iris.arff	No
	The tasks for independent test would be generated if the path is valid. More details are in the subsection Use the independent test dataset		

Moreover, there are some parameters that are not commonly used

OUTER_FOLD_TEST_FILES_SERIAL	Specify the test set for outer n-fold	./o1te,./o2te,..	No
	Separate the dataset into test sets based on the supplied serials, and the number of folds will be generated by the number of files. This parameter can not exist together with OUTER_FOLDS. More details are in the subsection Specify data for cross validation		
OUTER_FOLD_TRAIN_FILES_SERIAL	Specify the train set for outer n-fold	./o1tr,./o2tr,..	No
	PML will generate the training dataset by the complementation of the test data set if only the test files are supplied and vice versa. More details are in the subsection Specify data for cross validation		
INNER_FOLD_TEST_FILES_SERIAL	Specify the test set for inner n-fold	./i1te,./i2te,..	No
	Separate data into test sets based on the supplied serials, and the number of folds will be generated by the number of files. This parameter can not exist together with INNER_FOLDS. Note that files for inner folds should be the integer that is multiple of the number of outer folds. More details are in the subsection Specify data for cross validation		
INNER_FOLD_TRAIN_FILES_SERIAL	Specify the training set for inner n-fold	./i1tr,./i2tr,..	No
	PML will generate the training set by the complementation of the files in INNER_FOLD_TEST_FILES_SERIAL if only the test files are supplied and vice versa. More details are in the subsection Specify data for cross validation		
TEST_FILE	Arff file for test	./test_data	Depend on OUTER_FOLDS
	Supply test data alone, and need to set OUTER_FOLDS as 1. In this		

	case, cluster and variable selection will be passed		
TEST_FILE_SERIAL	Test file with serial	./test_serial	Depend on OUTER_FOLDS
	Generate test files by separating FILE with the supplied serial file. This parameter cannot exist together with TEST_FILE.		
MAX_MEMORY	Specify the maximum available memory of each task.	1g	No
	Default is 1g, and this parameter is only valid for WEKA task.		
CORE_NUM	Specify the number of CPU cores for PML	4	No
	The default is to use all the cores. For PML-Server, this parameter will only valid for the result analysis.		
COMPRESS_LEVER	The level when compress the 3rd-party programs	0	No
	Default is 9, and PML will only pack files when set the value as 0. This parameter will only valid for the 3rd-party programs (include waffles).		
RETRY	The limitation of retry times for one task	5	No
	Default is 3		
FEATURE_THRESHOLD	Select variables when its value is larger than the threshold	0.7	No
	Default is 0, and this parameter is only valid for the n-fold variable selection tasks. More details are in the subsection Some Modify Parameters for Dimension Reduction.		
SEED	The seed for pseudo random	1	No
	Default is 1		
SEED_CLU	The random seed for cluster	1	No
	The value will be the same with SEED by default.		
SEED_INNER	The random seed for inner folds validation	1	No
	The value will be the same with SEED by default.		
SEED_OUTER	The random seed for outer folds validation	1	No
	The value will be the same with SEED by default.		
SLEEP_TIME	The CPU sleep time after each monitoring	1	No
	More details are in the subsection Control the process of PML		
BOINC_OPTION	Options for BOINC configures		No
	This parameter is only valid for the PML-Server. More details are in the subsection Modify PML-Server		

TIME_LIMIT_ TASK	A deadline (in seconds) for running tasks	3000	No
	This parameter is only valid for PML-Desktop. If a task has been running exceed the time limit, the task would stop. More details are in the subsection Control the process of PML		
TIME_LIMIT_RESULT	A deadline (in seconds) for running tasks in result analysis	300	No
	If a task of result analysis has been running exceed the time limit, the task will stop. More details are in the subsection Control the process of PML		
TASK_COMPRESS_NUM	The number of tasks that would be packed into a 'large' task.	50	No
	Only valid for PML-Server. More details are in the subsection Packing the tasks		
TASK_DEVIDE_NUM	The number of parts which the tasks would be divided into.	3	No
	Only valid for PML-Server. More details are in the subsection Packing the tasks		
ONE_OUT_PER_OPT	Restrict the number of the methods from each parameter optimization to 1	1	No
	This parameter is provided in case of the situation that some methods would have more than one parameter which shows the best performance. If this parameter is valid, the sortable table in the part of Parameter Optimization would contain only one optimal parameter (or one optimal combination of parameters) from each method.		

Notes:

1. Some variable selection methods can not rank the variables, **FEATURE_OUT_FEATURES** would become invalid in this case. More details could be found in the subsection Some Parameters for Dimension Reduction.
2. The dataset for n-fold cross validation is generated with pseudo random number which can be modified by changing the **SEED**. Alternatively, users can specify the data for each fold according to respective requirement. Note that the inner-fold depends on the outer-fold, thus the number of files in INNER_FOLD_TRAIN/TEST_FILES_SERIAL should be the integral multiple of the number of OUTER_FOLDS. More details could be found in the subsection Specify data for cross validation
3. PML support the '#' as annotation in the input script. Note that the '#' should be located at the head of the line.
4. The result analysis of PML mainly focus on modeling, thus TT should be at the last of the parameter **STEP**.

4.3 Add machine learning methods to PML

(Note: This chapter could be passed if users do not need to add the methods immediately.)

PML has integrated amount of machine learning methods of WEKA and waffles. Users could find the information and modify the methods at the subfolder 'pml/config'. If users want to add some new methods or some existed methods with specified parameters, the suggestions below are available for reference.

4.3.1 Add WEKA Method

Users could add the methods of WEKA through the following steps:

1. Open WEKA explorer and load a data.
2. Choose the algorism which needs to be added and modify the parameter.
3. Right click the algorism and select 'Copy configuration to clipboard'.
4. Paste the detail into the related file (pml/config/cluster, pml/config/feature or pml/config/traintest) and then add **Serialnum.weka** in front of the added line. (The **Serialnum** here only needs to be different from the other methods)
5. Specially, the variable selection methods in WEKA have two parts: **Attribute Evaluator** and **Search Method**. Thus adding a variable selection method needs to paste the information of **Attribute Evaluator** first and then add
   -s "**Search Method**"

4.3.2 Integrate 3rd-party program

PML provides the command line interface to integrate the 3rd-party programs. Considering that most of the programs have the environments and interfaces themselves, thus PML has some requirements:

1. Make sure the program can identify the data of ARFF format. If the 3rd-party program does not support ARFF format, users need to add a script to convert the dataset into ARFF format. The name of the input data need to be data.arff for cluster or variable selection method and train.arff & test.arff for modeling method.
2. The name of output file needs to be out.txt, and the output file needs to fit for a template. All the templates are provided in '**<pml_root>/pml/examples/templates**'.
3. The related operating environment needs to be installed (like install JAVA JRE for WEKA).
4. A file named 'script' which contains the command lines that need to be typed into the terminal.

There are two examples for explaining how to add a program into PML:

Example 1

R provides a package named randomForest to use random forest for classification and regression. This example shows how to add the random forest method into PML:

1. Confirm that the R is installed successfully and can be called in command line window.
2. Copy **<pml_root>/pml/examples/R_rf** to **<pml_root>/pml/config/tt**.

3. Modify the file **<pml_root>/pml/config/traintest** by any test editor, and add a line
   8.3rd R_rf
   after the 7th method, then save the file.
4. (optional) Use the script reassign.pl in **<pml_root>/pml/config** to reassign the sequence number of the methods:
   perl reassign.pl traintest

PML can utilize the R_rf to do classify after the steps above. Besides, users can check the program by the steps as follows:

1. Copy **<pml_root>/pml/examples/R_rf** into any folders
2. Copy **<pml_root>/pml/examples/data/train.arff** and **<pml_root>/pml/examples/data/test.arff** into the same folder
3. Open a command line window, and make the work space under the folder (use 'cd' command), then type
   R -f rf.r

Example 2

The software SVMlight [Joachims, 1999] is an implementation of Support Vector Machines in C. To integrate SVMlight into PML, users need the steps below:

1. Copy **<pml_root>/pml/examples/svm_light** to **<pml_root>/pml/config/tt**.
2. Modify the file **<pml_root>/pml/config/traintest** by any test editor, and add a line
   8.3rd svm_light
   after the 7th method, then save the file.
3. (optional) Use the script reassign.pl in **<pml_root>/pml/config** to reassign the sequence number of the methods:
   perl reassign.pl traintest

Then SVMlight is available with the name 'svm_light'.

Some differences:

1. The file script can be found under the program folder or subfolders. The location of script in the *R_rf* is different from the scripts in the *svm_light* because of cross platform. The *R_rf* needs the installation of R, which can be called globally by setting the environment value, thus the files in the *R_rf* are only scripts. On the other hand, the *svm_light* needs to be compiled from source code, which leads the executable file only fits for the corresponded platform. It is not a matter when using PML-Desktop, but when to use PML-Server, the Client machines may have different platforms, thus the executable files need a mechanism for management. Therefore, PML provides these subfolders below to support multiple files for different platforms:

Subfolder Name	Platform
win	windows
win_x86	32 bit windows
win_x86_64	64 bit windows
linux	Linux
linux_x86	32 bit Linux
linux_x86_64	64 bit Linux

Users can move the files, which are conflict with each other by cross platform, into the respective subfolders. Then PML would detect the platform and copy the files from the corresponded subfolder into the root of the work path before executing a task. This might be useful in using PML-Server.

2. The copy sequence of the platform folder is
   win -- win_x86/win_x86_64
   or
   linux -- linux_x86/linux_x86_64
   Thus, users can avoid copying the same files into more than one subfolders. Moreover, not all the subfolders are necessary. It means that if 3rd-party program is not sensitive with 32 bit or 64 bit, the subfolders about x86/x86_64 is unnecessary.
3. Even though R_rf does not need to modify the subfolders for cross platform, it is necessary that each Client machine needs to install and configure R when using PML-Server.
4. If users compile the SVMlight and set an environment value for it, the files in the *svm_light* could be used in a similar way with the *R_rf*, which means that users can add some programs in different ways.

Templates

PML integrates the 3rd-party program by command line interface, and could identify the output files that fit for the templates. The templates could be found at **<pml_root>/pml/examples/templates**, and the descriptions are as follows.

The templates for clustering:

template_cluster1
A standard output format of WEKA. The first list represents the serial of instances, and the second list represents the clustered class of each instance.
template_cluster2
A simplified template. Only one list represents the clustered class of each instance. The serial is the same as the sequence of the input data.
template_cluster3
An ARFF format data file. Sometimes 3rd-party program generates the data file directly, and in this case, the dimensionality reduction of PML would be invalid. More details could be found at Some Modify Parameters for Dimension Reduction

Templates for variable selection:

template_VarSeletion1
A standard output format of WEKA, and the useful line is the line begins with 'Selected attributes: …'
template_VarSeletion2
A simplified template. Only retains the useful line of *template_VarSeletion1*.
template_VarSeletion3
A standard output format of WEKA, and this template lists the rank of the variables.
template_VarSeletion4

A simplified template. Only retains the useful part of *template_VarSeletion3*.
template_VarSeletion5
A standard output format of WEKA, and this template lists the output when n-fold validation is utilized. The variable would be selected finally if it has been selected more than a specified time n (n could be set by users through **FEATURE_THRESHOLD**, and more details could be found at Some Modify Parameters for Dimension Reduction).
template_VarSeletion6
A simplified template. Only retains the useful part of *template_VarSeletion5*.
template_VarSeletion7
An ARFF format data file. Sometimes 3rd-party program generates the data file directly, and in this case, the dimensionality reduction of PML would be invalid. More details could be found at Some Modify Parameters for Dimension Reduction

Templates for modeling:

template_TrainTest1
A standard classification output format of WEKA. The five lists in it are: instances, actual classes, predicted classes, errors, predictions
template_TrainTest2
A simplified template of *template_TrainTest1*. Only the list of predicted classes is retained.
template_TrainTest3
A standard regression output format of WEKA. The four lists in it are: instances, actual values, predicted values, errors
template_TrainTest4
A simplified template of *template_TrainTest3*. Only the list of predicted values is retained.

Users need to make sure the output file (out.txt) fits one of the format of the provided template above.

Besides, Waffles (Gashler, 2011) has been integrated into PML as the 3rd part program. Users could refer the way of the integrations in the folder **<pml_root>/pml/config** and its subfolders (clu/fea/tt).

Some Parameters for Dimension Reduction

Most Variable Selection algorisms might select variables through the two ways:

1. Rank the variables by some rules.
2. Only select variables by some rules.

If a method satisfies case 1, the parameter **FEATURE_OUT_FEATURES** will be valid and PML would reduce the number of variables by the settings of users. However, if the method satisfies case 2, PML will ignore the parameter **FEATURE_OUT_FEATURES** (that means all of the selected attributes would be used in the new dataset), and add the string '**_nr**' after the name of the generated dataset.

If a variable selection method of WEKA is used, PML will detect whether the method could

rank the variables. But for 3rd-party program, PML would recognize the method as it could rank the variables by default. Users can change it just by creating a file named **algprop** with the words **no rank** in it.

As mentioned in chapter templates, PML supports the output file in ARFF format. However, in this situation, PML could not reduce the dimension of the dataset because the necessary information is missing. Thus, users need to add the file **algprop** with the string **no rank** to avoid the redundancy computing.

Besides, the parameter **FEATURE_THRESHOLD** is designed for the n-fold variable selection. N-fold variable selection means separating data into n folds, then do variable selection for each fold. Selected variables will be counted in the output. Parameter **FEATURE_THRESHOLD** acts on the count. For example, if **FEATURE_THRESHOLD** is set as 0.7 and a 10-fold variable selection task is executed, only the variables that are selected at least 7 times will be used finally.

4.4 Parameter optimization

Many methods have parameters that could be set manually, and the comparison of the results depending on different parameters is important in modeling. PML supplies functions for parameter optimization, and user can utilize them as follows.

4.4.1 Parameter optimization for weka

For the methods in WEKA, PML supports this format:
MethodNum parameter values
For example, in pml/comfig/cluster, the first method is
1.weka weka.clusterers.SimpleKMeans -N 2 -A "weka.core.EuclideanDistance -R first-last" -I 500 -S 10
Here, the parameter '-N' means the number of clusters, and this line in the input script can set the parameter ranging from 2 to 4:
TT_ARG_OPT=1 –N 2:4
If parameter '-I' needs to be changed to 500, 600, 700 in the same time, the line will become to:
TT_ARG_OPT=1 –N 2:4 –I 500:100:700
',' could be used to separate different methods.
The table below shows the operators supported by PML in parameter optimization:

Operator	Example	Out Value
a:b	3:5	3 4 5
a:b:c	3:2:9	3 5 7 9
a^b:c	2^-2:2	0.25 0.5 1 2 4
a^b:c:d	2^-2:2:2	0.25 1 4
M;N	Ture;False	Ture False
M;N	3:5;7:2:13;yes	3 4 5 7 9 11 13 yes
_ON_	-P _ON_	Add a parameter (only for WEKA)
OFF	-P _OFF_	Delete a parameter

M**N**	P=****1;3:4	P=1 P=3 P=4
~~~	3~~~;5;7	3;5 7

Here, the operator ~~~ is to escape the operator to an ordinary character.

4.4.2 Parameter optimization for 3rd-party program

For 3rd-party program (including the methods of Waffles), the parameter optimization becomes more complicated due to the variant of the input formats. Thus, PML provides a relatively uniform format:

MethodNum Filename ###string### ###pattern### num ###ChangeDetail###

The **Filename** means each file being in the method folder. The **string** is the character string which needs to be changed. The **pattern** is used to split the **string** into some parts and the splited elements can be specified by **num** and changed to **ChangeDetail**.

Here is an example to explain the format. By default, the 19th modeling method is the 'waf_randomforest' that is a Random Forest algorism of waffles, and the related folder can be found in pml/config/tt. This line can be found in the file named 'script' of each subfolders:

waffles_learn train -seed 1 train.arff randomforest 50 > mod.json

The 50 here means the number of sub trees and it can be changed by adding this line into input script:

TT_ARG_OPT=19 script ###train.arff randomforest 50### ### ### 3 ###60:10:80###

This line is used for multiple parameters in one **string**:

ArgNum Filename ###string### ###pattern### num1;…;numn ###ChangeDetail1&&&…&&& ChangeDetailn###

It shows that use ';' to separate **num** and '&&&' to separate **ChangeDetail**.

Similarly, use ',' can separate different methods.

Notes:

1. PML will change all the files with the same **Filename** in different platform subfolders, and this would be useful for cross platform if users do not want to change the parameters due to different platforms.
2. The **string** in a script needs to be unique unless users want to change two places at the same time. However, it might be impossible in some cases, thus there are some tips that might be helpful in PML. For example, this line would appear in the scripts of R or Python.
   function(Var1,Var2,Var3);

   1)
   It's not hard if only **Var3** needs to be changed by setting **string** as
   function(Var1,Var2,Var3
   ,**pattern** as ',' and **num** as 3.

   2)
   If the Var1 needs to be changed, the **string** should be:
   Var1,Var2,Var3);

However, it might not be unique in a script. To avoid it, the annotations might be useful. This line could become unique by adding an annotation:

function(Var1,Var2,Var3);#annotation

Then the **string** will become:

Var1,Var2,Var3);#annotation

thus **Var1** can be changed without breaking the line.

3)

Another way is to use the operator '****'. If both **Var1** and **Var3** need to be changed (assume that the Var1 is expected to be 2:4 and Var3 to be 5;6), the **string** would be:

function(Var1,Var2,Var3);#annotation

Where the **num** is **1;3** and the **ChangeDetail** is:

function(**2:4&&&5;6****)~~~;#annotation**

4.5 Manage the machine learning methods

There are three files, *cluster*, *feature* and *traintest* in the folder **<pml_root>/pml/config**. The available machine learning methods are recorded in the three files. Users could use the methods by specifying the sequence number. For example, 'TT_ARG = 1,2,4' means that use the modeling methods with the sequence number '1', '2' and '4' in the file *traintest*. Users could add, remove and change the sequence numbers manually.

Besides, sometimes users might want to 'insert' a method with a small sequence number. The script **reassign.pl** in **<pml_root>/pml/config** would be useful. Assuming that the user wants to insert a method with sequence number '3' into the file *cluster*, and the content in *cluster* is like this:

1.weka …
2.weka …
3.weka …
…

Users could add a line under the line '2.weka …', then the content becomes as:

1.weka …
2.weka …
3.3rd …
3.weka …
…

Then open a terminal, make the work root as the **<pml_root>/pml/config** (use the 'cd' command), and use the **reassign.pl** with the command line:

perl reassign.pl cluster

Then the sequence numbers would be reassigned as follows:

1.weka …
2.weka …
3.3rd …
4.weka …
…

4.6 Use the independent test dataset

Sometimes users want to use the independent test dataset to verify the performance of the model. PML currently supports the independent test by the parameter **INDEPENDENT_DATA**. The **INDEPENDENT_DATA** is used to specify the test dataset, and the training dataset is the whole dataset for cross validation. If some dimension reduction tasks have been processed and the number of the attributes of the dataset has reduced, PML would reduce the related attributes in the independent test dataset before modeling.

It should be noted that there is a limitation of this function at present. For now, PML does not support the variable selection methods that would combine the attributes into new attributes (such as PCA or LSA in WEKA) because an additional model is necessary to process the attributes. This function will be improved with the update in the future.

4.7 Specify data for cross validation

PML could generate datasets for cross validation by the pseudorandom number, which could be modified by changing the random seed. However, sometimes users want to specify the datasets depending on some special requirements. In this case, several parameters were provided by PML: **OUTER_FOLD_TEST_FILES_SERIAL**, **OUTER_FOLD_TRAIN_FILES_SERIAL**, **INNER_FOLD_TEST_FILES_SERIAL** and **INNER_FOLD_TRAIN_FILES_SERIAL**.

The 'SERIAL' means the sequence numbers of the instances of a dataset. Moreover, two formats could also appear in one serial file:

1
2
3
or
1,2,3

PML would decide the outer folds and inner folds according to the serial files if users provided. Moreover, if only train files or test files are provided, PML will generate the test/train datasets by the rest of the serial numbers.

Besides, if the serial files for inner folds are provided, users need to confirm that the number of files for inner folds is the multiples of the number of outer folds. Assuming that inner fold is 3 and outer fold is 4, PML will generate the datasets for inner cross validation by this sequence:
I1O1 I2O1 I3O1 I1O2 I2O2 I3O2 I1O3 I2O3 I3O3 I1O4 I2O4 I3O4
Where the **IaOb** means the file for inner fold 'a' and outer fold 'b', thus users need to provide serial files according to the sequence.

4.8 Outputs

PML provides HTML outputs. A tree is provided, which records the details of the process of data. Users could view the details by the links provided by the tree and retry tasks according to the tips in the detail page. A links of the tree has one of the classes:

Name	Class
Cluster	Cluster
VarSelection	Variable Selection
ParaOptimization	Parameter Optimization
CrossValidation	Cross Validation
ModelingOut	Modeling for outer fold
ModelingIn	Modeling for inner fold
IndepTest	Modeling for independent test

The **ParaOptimization** records the results for all of the parameters of one modeling method, and users could compare the preference among the parameters. The **CrossValidation** contains the details of the outer and inner (if exist) cross validations to help users to check whether a modeling method is stable.

Besides, task name with blue color means that the task has been done successfully while the red one means failed. PML has saved all the input/output datasets and scripts for users to retry a task, check the results and so on. Note that all of the methods from WEKA and Waffles would get the same results by setting the same random seed when users retry them. However, some other methods using completely random number might get different outputs each time.

Moreover, PML provided a sortable table, which records all of the results of modeling methods. The evaluation criteria for modeling are different between classification and regression, thus the formulas and terminology explanations will be provided according to the type of data.

In conclusion, the output of the PML is designed for the convenience of checking and finding the best modeling method (sometimes it might be combined with the cluster and variable selection methods), so as to understand and read when using it.

4.9 Examples

In order to make users comprehend the functions more expediently, some simple examples are provided in pml/examples. The descriptions are as below:

1	single_method_classify/ single_method_regress
	Simplest script. Only one method of modeling is used.
2	single_method_PO_classify/ single_method_PO_regress
	Script with only one method of some changed parameters of modeling.
3	muti_method_classify/ muti_method_regress
	Script with two methods of modeling,
4	muti_method_PO_classify/ muti_method_PO_regress
	Script with two methods of some changed parameters of modeling.
5	muti_method_Clu_PO_classify/ muti_method_Clu_PO_regress
	Script with methods of cluster and modeling, and the changed parameters are provided for both cluster and modeling methods
6	muti_method_Fea_PO_classify/ muti_method_ Fea _PO_regress
	Script with methods of variable selection and modeling, and the changed parameters are provided for both variable selection and modeling methods
7	muti_method_Clu_Fea_PO_classify/ muti_method_Clu_Fea_PO_regress

	Script with methods of cluster, variable selection and modeling, and the sequence is cluster -- variable selection – modeling. The changed parameters are provided for all of the methods
8	muti_method_Fea_Clu_PO_classify/ muti_method_Fea_Clu_PO_regress
	Script with methods of cluster, variable selection and modeling, and the sequence is variable selection -- cluster – modeling. The changed parameters are provided for all of the methods
9	test_data_classify/ test_data_regress
	Script to demonstrate the use of the test data for modeling tasks. Note that the value of **OUTER_FOLDS** needs to be set as 1.
10	test_data_serial_classify/ test_data_serial_regress
	Script to demonstrate the use of serial file to generate test data for modeling tasks. Note that the value of **OUTER_FOLDS** needs to be set as 1.
11	train_outer_serial_classify/ train_outer_serial_regress
	Script to demonstrate the use of serial file with serial of test data to generate the train/test data for n-fold cross validation. The train data here will be generated by the remainder of the provided data.
12	train_outer_serial_classify/ train_outer_serial_regress
	Script to demonstrate the use of serial file with serial of train data to generate the train/test data for n-fold cross validation. The test data here will be generated by the remainder of the provided data.
13	test_inner_serial_classify/test_inner_serial_regress
	Script to demonstrate the use of serial file with serial of test data to generate the train/test data for inner n-fold cross validation. Note that the serial files for respective outer folds needs to be provided by parameter **OUTER_FOLD_TEST_FILES_SERIAL** or **OUTER_FOLD_TRAIN_FILES_SERIAL**.
14	train_inner_serial_classify/ train_inner_serial_regress
	Script to demonstrate the use of serial file with serial of train data to generate the train/test data for inner n-fold cross validation. Note that the serial files for respective outer folds needs to be provided by **OUTER_FOLD_TEST_FILES_SERIAL** or **OUTER_FOLD_TRAIN_FILES_SERIAL**.

4.10 Recover and Reset an Experiment

PML allows users to recover the process through typing in the same command line again when the machine running PML crashed under some unexpected situations. For example, if the experiment **muti_method_PO_classify** with script file **pml/examples/muti_method_PO_classify** needs recovery. Users only need to type:

pml_desktop.pl pml/examples/muti_method_PO_classify

On the other hand, an experiment might be required to restart by some reasons. In this case, the option **--reset** is useful. With **--reset**, PML will delete all of the files related to the specified experiment. For example, users could restart the experiment **muti_method_PO_classify**, as mentioned before, by this command line:

pml_desktop.pl pml/examples/muti_method_PO_classify --reset

Note that when using pml_desktop, the option **--reset** would only delete the result folder of the experiment because of the simple relationships among the tasks. However, pml_server uses more than one machine to do computing. The task on a Client need the machine to receive the stopping messages from the Server before the cancel, and the related records in database (MySQL) need to be wiped. Therefore, it might take much time to reset an uncompleted experiment with PML-Server.

5. Some operating suggests

There are some suggestions that might be helpful when using PML. Users could refer them as needed.

5.1 Make input script more readable

An input script is necessary for an experiment, and it is not hard for users to write it. However, some lines in the script might be too long to read by users in some situations. For example, if 3 modeling methods are used and need to do parameter optimization, the line of parameter **TT_ARG_OPT** might be as this form:

TT_ARG_OPT = 1 -A 5:8 , 4 file ###a,b,c### ###,### 2 ###10:10:50### , 7 -C 2^-3:3

To make the line more readable, PML allows users to write the parameters in separate lines, and these lines below are equivalent to the line above:

TT_ARG_OPT = 1 -A 5:8 , 7 -C 2^-3:3

TT_ARG_OPT = 4 file ###a,b,c### ###,### 2 ###10:10:50###

The optimization details of one method are allowed to be written in any lines, but note that writing details of one method into two lines would not be supported.

The supported parameters are listed in the table:

CLUSTER_ARG	CLUSTER_OUT_INSTANCES	CLUSTER_ARG_OPT
FEATURE_SELECT_ARG	FEATURE_OUT_FEATURES	FEATURE_ARG_OPT
TT_ARG		TT_ARG_OPT
INNER_FOLD_TRAIN_FILES_SERIAL		INNER_FOLD_TEST_FILES_SERIAL
OUTER_FOLD_TRAIN_FILES_SERIAL		OUTER_FOLD_TEST_FILES_SERIAL

5.2 Control the process of PML

PML would monitor the statues of tasks until all of them are completed. The parameter **SLEEP_TIME** in seconds is used to control the speed of the monitoring, and the default value is 0 for PML-Desktop and 1 for PML-Server. PML would sleep the specified seconds after each time of monitoring. The higher the value was set, the more computing source for tasks. On the other hand, the lower value was set, the quicker speed for the main process. Therefore, if a large scale

of data (like 10000 x 500) is used, it is better that set the value of **SLEEP_TIME** as 1 or more. Specially, when PML-SERVER is utilized, the machine of server is only used to send and receive tasks, thus the SLEEP_TIME is suggested to be set as 1 or more.

Sometimes 3rd-party software might not stop running by some reasons such as endless loop. To avoid the impact from the endless task, PML provides a limitation for each tasks. The parameters are **TIME_LIMIT_TASK** (for PML-Desktop), **BOINC_OPTION** (for PML-Server) and **TIME_LIMIT_RESULT**. When using PML-Desktop, users can use **TIME_LIMIT_TASK** to set the limitation for each task, the unit is in seconds and the default value is 86400 (1 day). When the run time of a task exceeds the limitation, the task would be regarded as an error task. The same function can be achieved by set the parameter **BOINC_OPTION** as:

BOINC_OPTION = --delay_bound x

The **x** here is the value of time limitation (default is also 86400). The parameter **TIME_LIMIT_RESULT** is the time limitation of result analysis, and the default is 300 (5 minutes). It is set for avoiding unexpected system fault which lead to the halt of a task.

5.3 Configure PML-Server

PML-Server only uses the functions of the BOINC-Server by command line interface without modifying the code of it, thus the sets of the BOINC-Server is still valid. To make PML more fit for BOINC, some configures are changed after executing the script PML2BOINC.pl. Moreover, Users could also modify and change the BOINC configures through the official guide from:

http://boinc.berkeley.edu/trac/wiki/ServerIntro

and

http://boinc.berkeley.edu/trac/wiki/ProjectConfigFile

In addition, some parameters can be modified directly by using the **BOINC_OPTION** parameter in the input script. The supported functions are as below:

Name	Description	Default
--delay_bound	An upper bound of the time consuming of a task	86400 (1 day)
--rsc_fpops_est	An estimate of the number of floating-point operations required to complete the task	3600e9 (3600 GFLOP)
--rsc_fpops_bound	An upper bound of the number of floating-point operations required to complete the task	86400e9 (86400 GFLOP)
--rsc_memory_bound	An estimate of tasks' largest memory used size.	5e8 (about 512MB)
--rsc_disk_bound	A bound of the maximum disk space used by the task	1e9 (about 1GB)
--max_error_results	Task would stop if the errors exceed the set value	3

The parameters **--rsc_fpops_est** and **--rsc_fpops_bound** are provided by BOINC to reduce the over computing of a task. However, if a method choosed by users takes too much time, the --

rsc_fpops_bound might be set as a larger value, 86400e11 for instance.

The parameters **--rsc_memory_bound** and **--rsc_disk_bound** are provided to filter the Clients. The Client machine would not be sent task if the memory size (RAM size) or free disk space is lower than the respective bound.

More detailed descriptions could be found from http://boinc.berkeley.edu/trac/wiki/JobIn

Besides, BOINC is designed to support the volunteer computing, but PML-Server only uses it for grid computing, thus the hardware demand of the Server is not so high. If the number of Clients is less than 1000, a machine with a Pentium IV CPU would be enough to satisfy the requirement. This estimation is referenced from the guide of SZTAKI (http://desktopgrid.hu/index.php?page=23), a project using BOINC to achieve grid computing.

5.4 Manage Clients Remotely

PML services as a project in BOINC-Server, thus users need to update the project manually (click the update button only once) if they want Clients to receive the tasks immediately. However, this action might be inefficient if the number of Clients is large. Therefore, the remote management would be useful for this situation. Users could use one machine installed with BOINC-Client to control all of the Clients by simple settings. For example, if the ip address of the manage machine is 192.168.0.10, then for each Client, these steps are needed:

1. Create a file named **remote_hosts.cfg** (records the ips of manage machines) and with this line in it:
   192.168.0.10
2. Create a file named **gui_rpc_auth.cfg** (records the password of this Client) and input a key in it, like:
   123456
3. Copy the two files into boinc directory. For Windows, the path is:
   C:\Documents and Settings\All Users\Application Data\BOINC
   and for Linux is:
   /var/lib/boinc-client
4. Restart the boinc-client on the Client or restart the machine.

After the configurations, Clients could be remotely controlled by **boinccmd** with this command:
boinccmd --host <host_ip> --passwd <password> [commands]
The **<host_ip>** is the ip of the Client whice needs to be controlled, and the **<password>** is the key in the file **gui_rpc_auth.cfg**. For example, if a Client (with ip 192.168.0.11) needs to update project (with ip http://192.168.0.1/pml), the command line below will be useful:
boinccmd --host 192.168.0.11 --passwd 123456 --project http://192.168.0.1/pml update

More details could be found from:
http://boinc.berkeley.edu/wiki/Controlling_BOINC_remotely
http://boinc.berkeley.edu/wiki/Boinccmd

5.5 Packing the tasks

PML-Server uses the BOINC-Server to transmit the tasks to the Client machines over the network. Compared with PML-Desktop, PML-server would cost more time, about 3-5 seconds, for data transmission and task initialization. Thus for the task with a small scaled dataset, the execution time would be less than the initialization time and often causes low utilization of the computing resources.

In order to improve the efficiency for small data, the tasks would be packed and sent together as some 'large' tasks to the Client machines. The number of the tasks in a 'large' task is decided by the simple way:

$$\mathrm{EstNum} = \frac{30000}{InstancesNum \times AttributesNum}$$

$$\mathrm{CompressNum} = \min\left\{\mathrm{EstNum}, \left[\frac{TotalNum}{DevideNum}\right]\right\}$$

$$\mathrm{CompressNum} = 1 \text{ if } \mathrm{CompressNum} < 1$$

Where the $\mathrm{CompressNum}$ means the number of 'small' tasks in a 'large' task. The $TotalNum$ means the number of the tasks that waiting for execution. The $InstancesNum$ and $AttributesNum$ mean the number of instances and attributes of the dataset. The $DevideNum$ is 1 by default.

Since the number of the initialized tasks changing all the time in the process, the EstNum would be larger than the number of all the executable tasks, thus the $DevideNum$ is needed to ensure that the tasks would separate into some parts.

The EstNum and $DevideNum$ could be set by modifying the values of **TASK_COMPRESS_NUM** and **TASK_DEVIDE_NUM** in the input script.

6. Development and testing environment

PML was developed on Ubuntu 12.0.4 LST. The SDK was Eclipse SDK 4.2.2 with the plugin EPIC 0.6.47. The version of PERL was 5.14.2.

PML was developed to support Windows and Linux, and these platforms have been utilized for testing PML:

For PML-Desktop, the test platforms are: Windows 7 x86/x64, Windows 8 x86/x64, Ubuntu 12.04 LST x86/x64, Debian 7.1.0 x86/x64, Fedora 19 x86/x64, and CentOS 6.4 x86/x64.

For PML-Server, the test platforms for SERVER (the machine on which the BOINC-Server installed) are: Ubuntu 12.04 LST x64, Debian 7.1.0 x64, Fedora 19 x64 and CentOS 6.4 x64, and for CLIENT, the platforms are the same as that for PML-Desktop.

The versions of PERL in the platforms are different. In Linux platforms, the versions are the related ones provided by the system. In Windows, the PERL is provided by ActivePerl 5.16.2.1602 or StarwberryPerl 5.16.2.1 when tested.

7. Conclusion

PML is free software aiming at helping users to save time in modeling and the related work. Now, PML has mainly supported dimensionality reduction (cluster and variable selection), parameter optimization and cross validation, and could process the tasks in parallel with one or more machines.

We sincerely hope that this manual would be helpful. We look forward to making improvements to PML based on community fee dback. If you enjoy the software and find it helpful; have a question; want to report a bug; or want to suggest a new feature, please send your emails to liml@scu.edu.cn.