User's Guide to JGrid

Ludwig-Maximilians-Universität München
Lehr- und Forschungseinheit für Datenbanksysteme
Oettingenstraße 67
80538 München
Germany

INSTALLATION GUIDE

The distributed source code has been written in Java Version 6.
Java has to be installed on all client machines and the server machine.
Unzip the file “jgrid.zip” to a directory on the server machine and copy the file “jgrid.jar” to all client machines.

Starting the server:

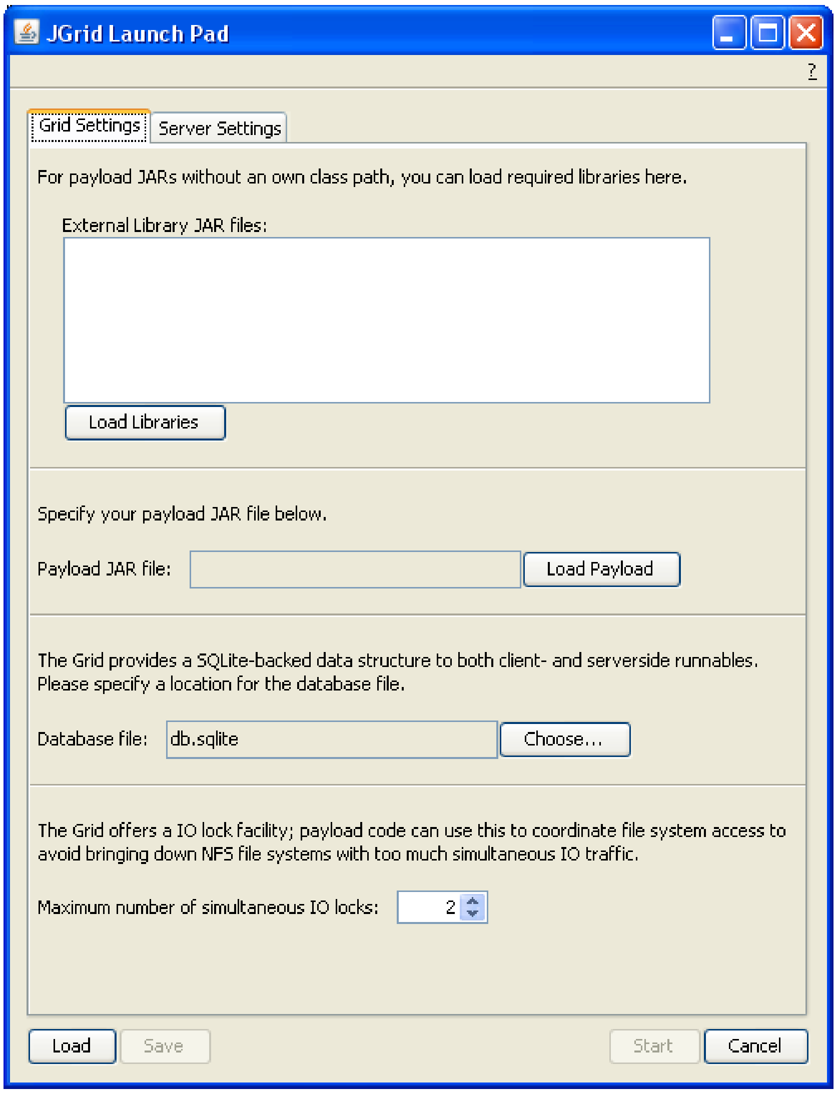
On the server machine the user first starts the application jgrid.jar without parameters

java –jar –jgrid.jar

or with -cfg <gridcfg file> to load a previously saved configuration from the specified file.

java –jar – jgrid.jar –cfg<configFilename>

This can also be done by starting JGrid without parameters and clicking on the “Load” button in the lower left corner.



For payload JARs without an own class path, you can load required libraries here.

External Library JAR files:

Specify your payload JAR file below.

Payload JAR file:

The Grid provides a SQLite-backed data structure to both client- and serverside runnables. Please specify a location for the database file.

Database file: db.sqlite

The Grid offers a IO lock facility; payload code can use this to coordinate file system access to avoid bringing down NFS file systems with too much simultaneous IO traffic.

Maximum number of simultaneous IO locks: 2

Starting the clients:

The JGrid JAR file needs to be started on all client machines. External JAR libraries, the payload code etc. do not need to be present on the client machines as they are being distributed over the network by the server.

There are two ways to start the client on the client machine:

- Have it seek the server and connect automatically; this only works if server and client are on the same local subnet:

  java –Xmx<memory> –jar – jgrid.jar -client -seek <port>

  with <port> being the broadcast port (which is the server port incremented by one.) and <memory> being the memory used for the computation; e.g. 4G
- Directly connect to a server. This works across NAT and firewalls as long as the client machines can reach the server machine on the server port (i.e. port forwarding or similar measures are available for this port and machine.)

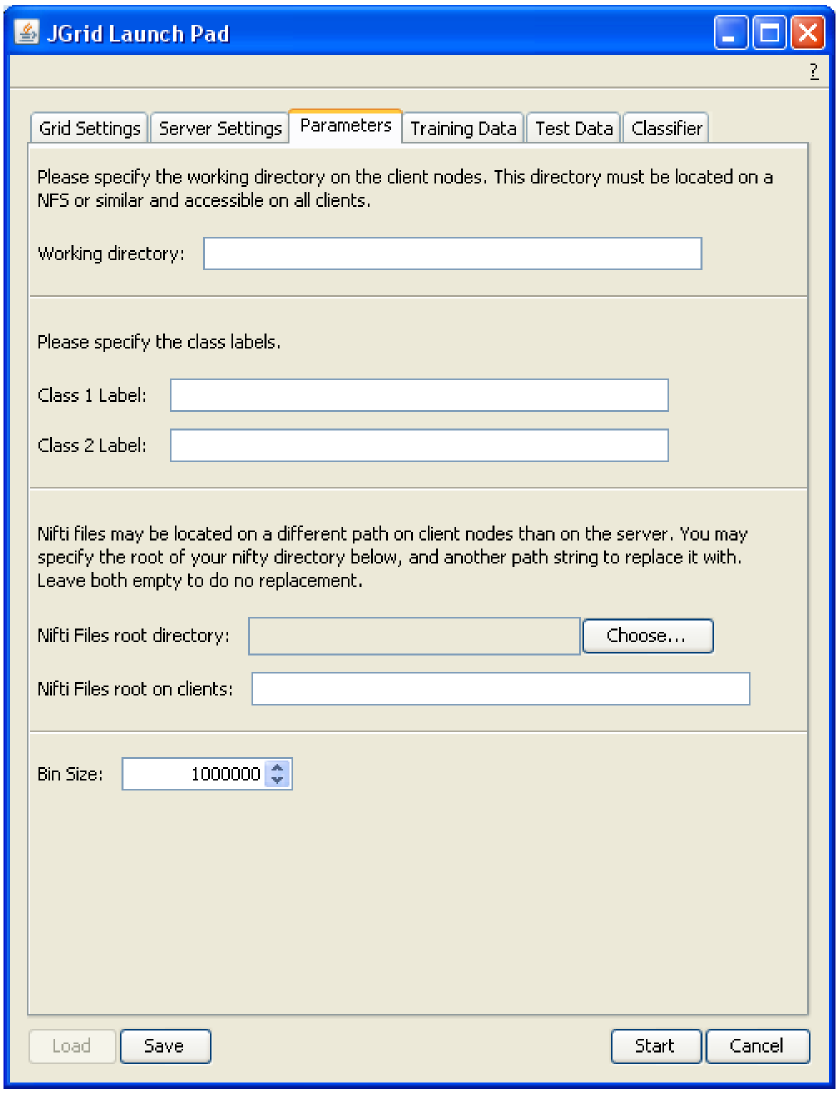
  java –Xmx<memory> –jar – jgrid.jar -client <server host> <port>

  specifying the server's IP address or host name and the port it is listening on (not the broadcast port.)

Configurations:

To use the FCC-Framework the payload application JAR file (payload.jar) needs to be loaded using the file selector “Payload JAR file”.
Standard JGrid options available through the GUI include the file location SQLite will use to save the database, the maximum concurrent number of file system I/O locks, the server port and whether to broadcast the server location in the local subnet, allowing clients in seek mode to autoconnect.

Once the “Payload JAR file” is loaded, several tabs for specifying the parameters for the FCC-Framework are displayed.



Please specify the working directory on the client nodes. This directory must be located on a NFS or similar and accessible on all clients.

Working directory:

Please specify the class labels.

Class 1 Label:

Class 2 Label:

Nifti files may be located on a different path on client nodes than on the server. You may specify the root of your nifty directory below, and another path string to replace it with. Leave both empty to do no replacement.

Nifti Files root directory:

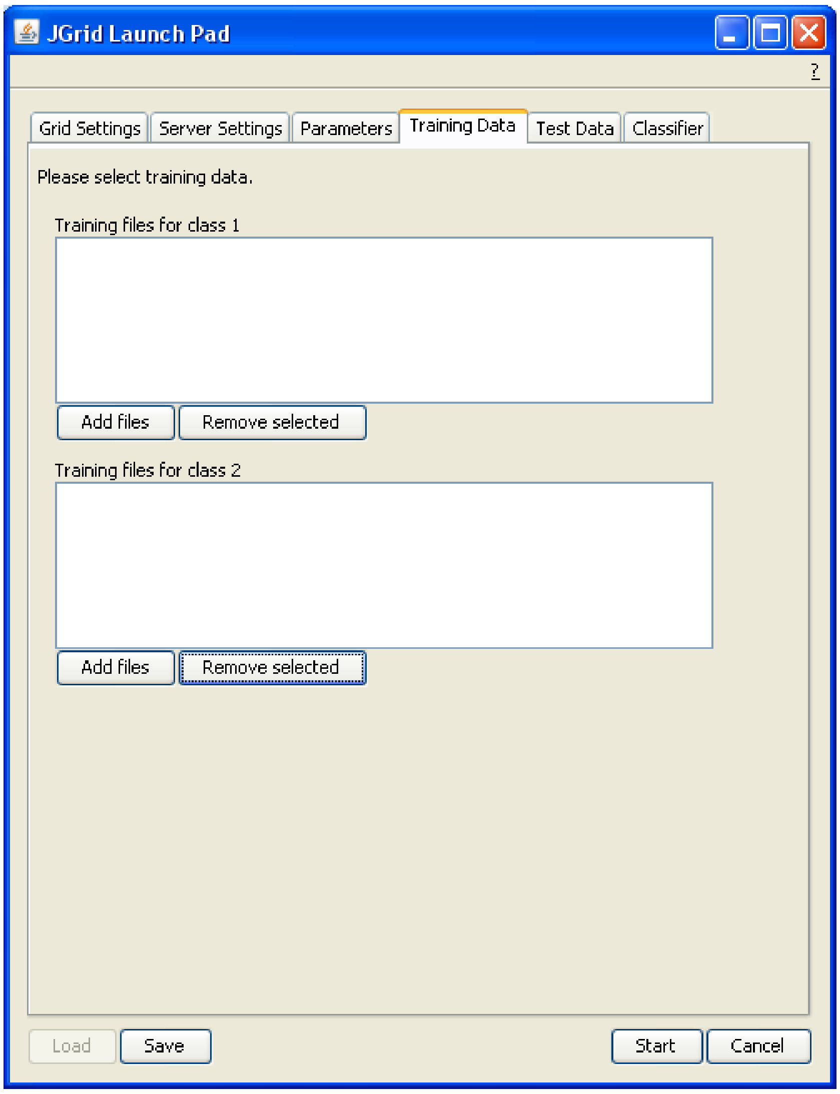
Nifti Files root on clients:

Bin Size: 1000000

These parameters include:

- **working directory**: directory for the output files.
- **Class label 1 and 2**: class labels of the different images, respectively.
- **Files root directory** (optional): Converts directory-path from different operating systems if the client is running on a Windows machine and the files are located on a Linux machine for example; e.g. "Nifti Files root on clients" is "N:\nifti\" but the files are located on the remote computer at "/home/usr/nifti/".
- **Bin size**: The Bin size has to be chosen according to the RAM used for the computation; the larger the Bin size the faster the computation

In the "Training Data" tab please specify the training files for each class.

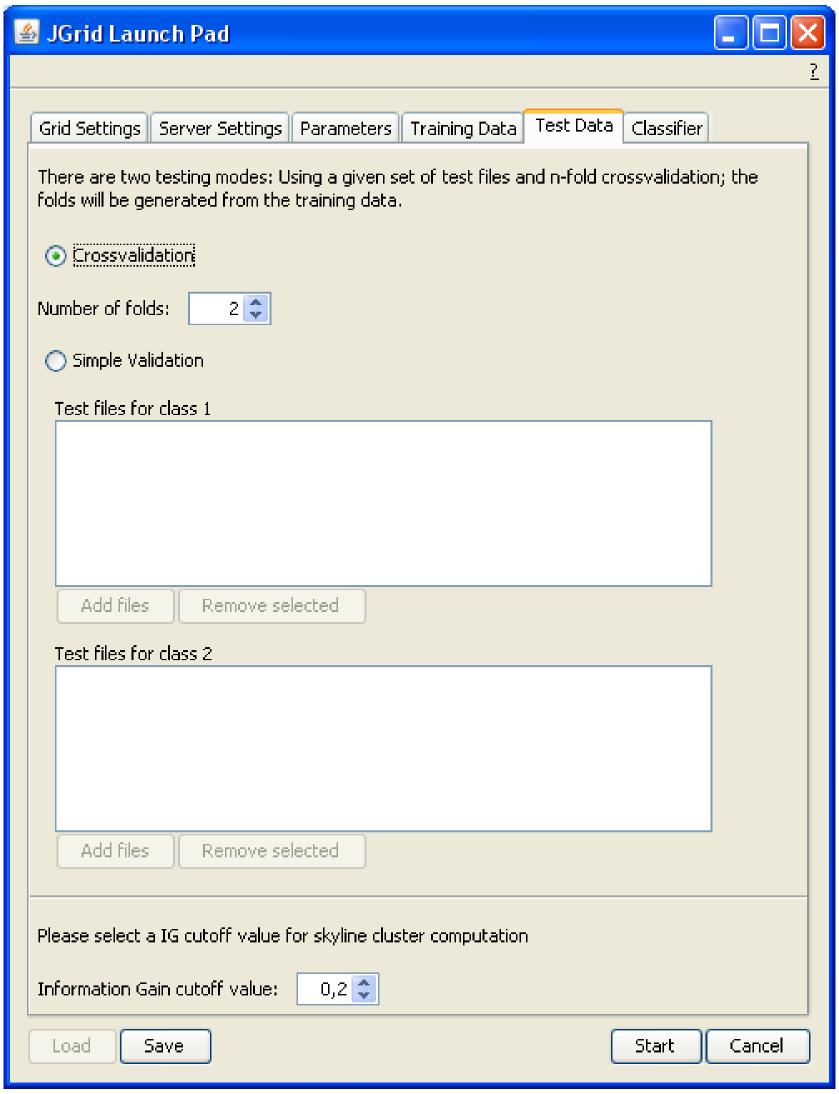

JGrid Launch Pad

Please select training data.

Training files for class 1

Training files for class 2

In the tab “Test data” the validation used for classification can be selected. If simple validation is selected the test files for each class have to be loaded.



There are two testing modes: Using a given set of test files and n-fold crossvalidation; the folds will be generated from the training data.

(●) Crossvalidation

Number of folds: 2

( ) Simple Validation

Test files for class 1

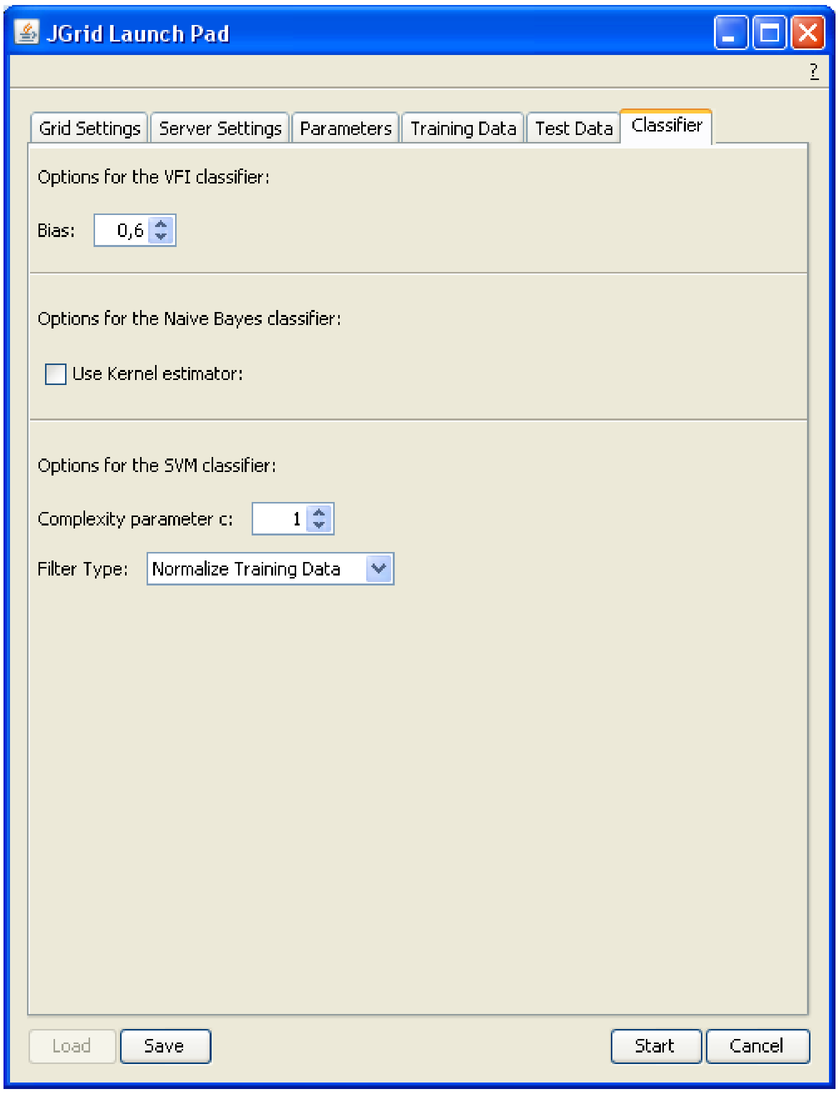
Test files for class 2

Please select a IG cutoff value for skyline cluster computation

Information Gain cutoff value: 0,2

For the computation of the skyline clusters an IG-cutoff value has to be chosen. The Cartesian coordinates and the Talairach coordinates for each voxel in a skyline cluster with an IG larger than IG cutoff are determined.

The parameters for the different classifiers can be selected in the “Classifier” tab:



Options for the VFI classifier:

Bias: 0,6

Options for the Naive Bayes classifier:

☐ Use Kernel estimator:

Options for the SVM classifier:

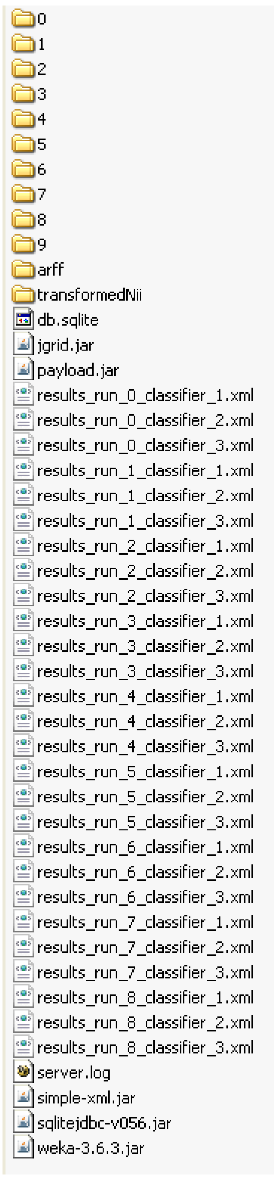
Complexity parameter c: 1

Filter Type: Normalize Training Data

The actual configuration can be saved with the “Save” button and reloaded with the “Load” button.

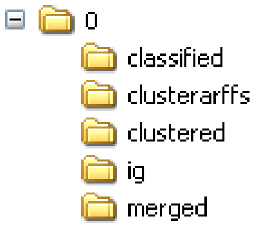
Application output.

Following files and folders are created in the working directory on the server machine:



```
0
1
2
3
4
5
6
7
8
9
arff
transformedNii
db.sqlite
jgrid.jar
payload.jar
results_run_0_classifier_1.xml
results_run_0_classifier_2.xml
results_run_0_classifier_3.xml
results_run_1_classifier_1.xml
results_run_1_classifier_2.xml
results_run_1_classifier_3.xml
results_run_2_classifier_1.xml
results_run_2_classifier_2.xml
results_run_2_classifier_3.xml
results_run_3_classifier_1.xml
results_run_3_classifier_2.xml
results_run_3_classifier_3.xml
results_run_4_classifier_1.xml
results_run_4_classifier_2.xml
results_run_4_classifier_3.xml
results_run_5_classifier_1.xml
results_run_5_classifier_2.xml
results_run_5_classifier_3.xml
results_run_6_classifier_1.xml
results_run_6_classifier_2.xml
results_run_6_classifier_3.xml
results_run_7_classifier_1.xml
results_run_7_classifier_2.xml
results_run_7_classifier_3.xml
results_run_8_classifier_1.xml
results_run_8_classifier_2.xml
results_run_8_classifier_3.xml
server.log
simple-xml.jar
sqlitejdbc-v056.jar
weka-3.6.3.jar
```

For each fold one folder is created which contains following subfolders:



- **classified**: results of all three classifiers for that fold.
- **clusteredarffs**: internal preprocessing files necessary for the algorithm
- **clusterd**:contains the following files:
  - *clusterStatistics_merged.tab.txt:* statistics for each cluster.
  - *index_clusters_merged.tab* the location: (slice,row,column) and cluster-Id for each voxel.
  - *coordinates_cluster<clusterID>.txt:Cartesian* coordinates(row, col, slice) for each voxel in a skyline cluster with an IG value larger than IG-cutoff
  - *talairachCoordinates_cluster<clusterID>.txt:* Talairach coordinates(row, col, slice) for each voxel in a skyline cluster with an IG value larger than IG-cutoff
- **ig**: Information Gain value for each voxel. The number of files in that folder depends on the Bin size
- **merged**: Information Gain value for each voxel merged into one file.

The folder “transformedNii” contains files with the voxel information of the nii-files converted to “.txt” format. The “arff” folder contains the voxel information in arff-format.
In order to run the FCC-framework with different settings do not delete the folder “arff”. JGrid does skip the conversion of the nii-files in this case.

For each classifier and each fold the classification results are written into a xml-file in the working directory on the server machine. Logging information is written to the log file server.log in the working directory on the server machine.