Chapter 1

A brief introduction to MOA-IncMine extension

1.1 IncMine within the MOA environment

Let us now see how IncMine can be used within the MOA environment. MOA allows two run task either in command line or via its GUI. In order to run MOA with our frequent itemset mining extension, the library `IncMine.jar` must be included into the library path of MOA. For example, if we want to lunch the MOA GUI with our library simply have to edit the file `moa.bat` in the following way:

```
java -cp IncMine.jar;moa.jar -Javaagent:sizeofag.jar moa.gui.GUI
```

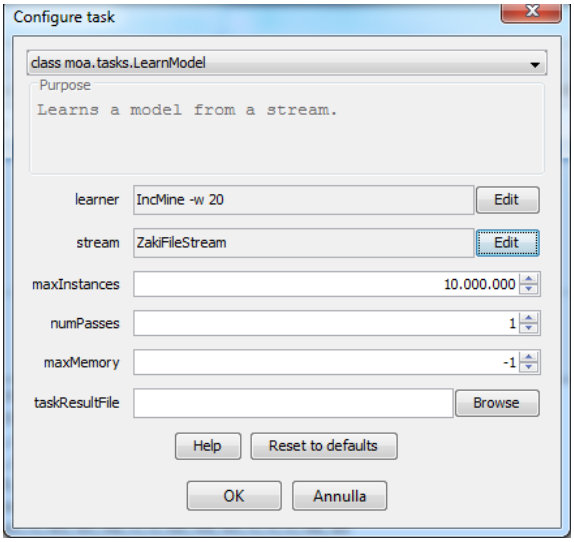
This command will start the graphic interface of the MOA environment. To run IncMine we have to use the Classification tab, and select task `moa.task.LearnModel`, or `moa.task.LearnEvaluateModel` if we also want performance evaluation.

Figure 1.1a shows an screen sample of `moa.task.LearnModel` configuration. We can set here the type of learner, the stream in input, the maximum number of transactions to be processed, the number of times the stream have to be passed (usually set to 1) and the maximum amount of memory desired. Figure 1.1b shows how the properties of IncMine algorithm can be set via user interface.

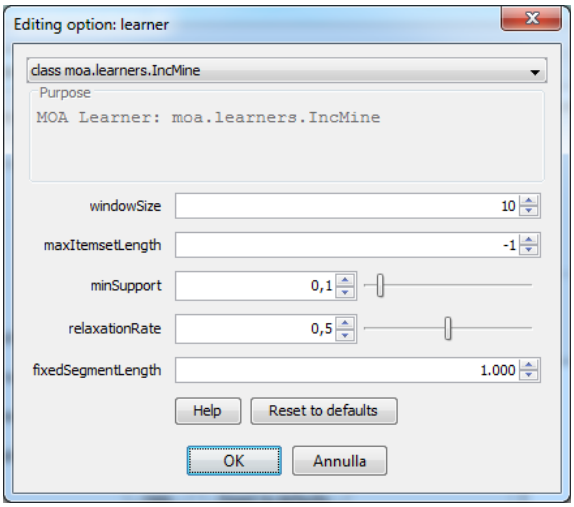
We can configure the learner and launch MOA also via command line. Consider the following example:

```
java -cp ncMine.jar;moa.jar -Javaagent:sizeofag.jar moa.DoTask ‘‘LearnModel
-m 100000 -l (IncMine -w 20 -m 5 -s 0.05 -r 0.4 -l 5000) -s (ZakiFileStream
-f T40I10D100K.ascii)’’
```

It runs the `LearnModel` procedure via a call to `moa.DoTask`. Each option



(a) LearnModel configuration.



(b) IncMine configuration.

Figure 1.1: Edit settings within the MOA GUI.

defined into a MOA class has a letter associated. The syntax to assign a value to an option is *ClassName [-opt value]*. Table 1.1 shows the options that are defined in the classes of our project.

1.2 Example of client class

Listing 1.1 shows a Java coded sample. It does not uses classes `LearnModel` or `LearnEvaluateModel` in this sample code, but the way it analyzes the

opt	Option
s	`minSupportOption`
r	`relaxationRateOption`
l	`fixedSegmentLengthOption`
w	`windowSizeOption`
m	`maxItemsetLengthOption`

(a) Class `IncMine`.

opt	Option
f	`zakiFileOption`

(b) Class `ZakiFileStream`.

opt	Option
l	`learnerOption`
s	`streamOption`
m	`maxInstancesOption`
p	`numPassesOption`
b	`maxMemoryOption`

(c) Class `LearnModel`.

opt	Option
l	`learnerOption`
s	`streamOption`
e	`evaluatorOption`
i	`instanceLimit`
t	`timeLimitOption`
f	`sampleFrequencyOption`
b	`maxMemoryOption`
q	`memCheckFrequencyOption`
d	`dumpFileOption`

(d) Class `LearnEvaluateModel`.

Table 1.1: Options of the classes defined in IncMine library.

stream is essentially the same that these classes do.

Firstly a new `ZakiStreamReader` is created and instantiated. It reads transactions from file `T40I10D100K.ascii` and provides an *iterator-like* interface to get them. While the stream has more instances (condition that can be checked via the method `hasMoreInstances()`), one can get the next transaction by calling method `nextInstance()` of the stream object.

In line 9 we create and instantiate the IncMine learner object. Then we configure the settings of the mining procedure (Lines 12-17). In MOA this is done by providing a value to some public options that are defined within the learner class, accordingly to the type of value that they are supposed to accept. In our specific case, IncMine allows to specify a value for

- `minSupportOption`, i.e., the Minimum Support Threshold σ. It allows values in $(0, 1]$, with default value 0.1.
- `relaxationRateOption`, i.e., the Relaxation Rate r. It allows values in $[0, 1]$, with default value 0.5.
- `fixedSegmentLengthOption`, i.e., the fixed-length of each segment. Its default value is equal to 1000.

- `windowSizeOption`, i.e, the number of segments that will be kept into the sliding window. It also corresponds to the size of the support vector associated to each SemiFCI and to the maximum k value allowed by the progressively increasing MST function $minsup()$. It has a default value of 10.

- `maxItemsetLengthOption`, i.e., the maximum length of itemsets to be considered. It can be useful set it to speed up the mining process when one knows the maximum size of the itemset of interest, because all itemsets in the current segment with length greater that this limit are discarded and do not influence the update process. It can be disabled by setting its value to -1. In this case the algorithm will perform a full mining of frequent closed itemset.

Listing 1.1: Compute FCIs over a data stream using IncMine.

```
import moa.learners.IncMine;
import moa.streams.ZakiFileStream;

public class Main {

    public static void main(String args[]){
        //read the stream T40I10D100K.dat
        ZakiFileStream stream = new ZakiFileStream(``T40I10D100K.ascii'')
        IncMine learner = new IncMine(); //create the learner

        //configure the learner
        learner.minSupportOption.setValue(0.01d);
        learner.relaxationRateOption.setValue(0.5d);
        learner.fixedSegmentLengthOption.setValue(1000);
        learner.windowSizeOption.setValue(20);
        learner.maxItemsetLengthOption.setValue(-1);
        learner.resetLearning();

        //prepare the stream for reading
        stream.prepareForUse();

        while(stream.hasMoreInstances() ){
            //pass the next instance to the learner
            learner.trainOnInstance(stream.nextInstance());
        }
        //output the final set of SemiFCIs
        System.out.println(learner);
    }
```