`PSI-Miner` Roadmap
Documentation of Code Development and Feature Enhancements

Antonio A. Bruto da Costa

1 Introduction

This report documents the development, versioning and feature enhancement plan for PSI-Miner, the Prefix Sequence Inference Miner. Other things to note:

- PSI-Miner uses Semantic versioning (`https://semver.org/`) for tracking release version numbers.
- I use tabs (not spaces) for indentation. In general 1 tab (one char) = 4 spaces (4 chars). It makes more sense to me to do it this way. One character in place of four or more characters, and hence smaller file sizes.

The code for PSI-Miner is maintained on GitHub at `https://github.com/antoniobruto/PSIMiner`. The most recent release of PSI-Miner is 1.0.0-rc1 and is available at `https://github.com/antoniobruto/PSIMiner/releases/tag/v1.0.0-rc1`. The list of all releases are available `https://github.com/antoniobruto/PSIMiner/releases`.

2 Version History

The following in a list of current and past versions, list from most recent to oldest.

- **v2.1:** ***Feature Enhancement: Class biased learning, Strict Pseudo-Targets***
  - **Class Bias during learning:** It is now possible to pass one or two parameters to psiMiner, the parameters being the configuration file and the second (optional) parameter being the bias. Learning explanations for a target demands learning explanations for both target class forms, with a positive or negative target literal appearing in the consequent. Under data bias, it is possible that for one form of the literal, the one in the minority, explanations are not learned. In this enhancement, we can bias learning to learn a predetermined literal form, even if it is in the minority within the data. This is the case especially when learning unsafe situations, which occur rarely. A positive bias, such as "+1" indicating learning biased towards the positive literal class, while a negative bias, such as "-1" indicates learning bias towards the negative literal class.

  - **Strict Pseudo-Targets:** For delay intervals of the form ##[0:k] appearing in the template, in specific cases, learning repetitive patterns of the form A ##[0:k] A can be impossible because the '0' in the delay can result in an idempotent forward influence relation. This means that the interval set for A is equivalent to the forward influence or end-match for A ##[0:k] A, resulting in an apparent stagnant zero gain. The reason is that the '0' in the delay interval doesn't change the left corner of intervals in the interval set for A. Taking the intersection of I(A)+[0:k] with the intervals in I(A) results in the same I(A). Without the ability to constrain I(A), the unchanging interval set manifests as a zero change in gain. With strict pseudo-targets the property template is adapted to use delay intervals of the form ##[k:k], thus changing the was end-matches and pseudo-targets are computed. This allows us to learn a richer set of properties.

- **v2.0:** ***Difference Mining***
  Properties are learned to establish how sets of traces are different. The aim is to learn properties that explain how sets of traces are different from each other. The tool configuration API has changed to allow a list of targets to be directly specified in the configuration file.

- **v1.0:** ***Learning Prefix Causal Sequences from User Knowledge***
  - Mining over multiple trace-files.
  - Support for user provided predicates (as Boolean expressions in conjunctive normal form)
  - The interface is terminal-based, and inputs/parameters/meta-parameters are specified via a single configuration file.

3 Version Release Plan

The following feature release(s) have been planned.

- **v2.2:** ***Multi-Trace support for learning predicates***
  A property is constructed using two elements, timing elements (interpreted as time delays), and Boolean expressions constructed from predicates of the form $x \sim c$, $\sim \in \{=, \leq\}$. It is possible to learn these predicates using local search heuristic algorithms. This feature enables such learning considering all traces.

4 Open Problems and Future Enhancements

In this section is a list of open problems, scientific and engineering, that we hope to solve/add to the PSI-Miner tool framework.

4.1 Research Problems

The following is a set of open research problems. It is unclear how much work each problem would require.

- ***Time Semantics:*** A significant attribute of the cause (antecedent) is the relative timing of events making up the cause with respect to the consequent. However, in time, it is possible

to create a variety of associations. For instance, continuous repetition using the semantics of *until*, first-match semantics, any-match semantics (presently implemented) and so on.

- ***Learning Complex Predicates:*** Beyond simple distance predicates $x \sim c$, $\sim \in \{=, \leq\}$, it would be interesting to broaden the predicate space and express more complex attributes, such as summary attributes like sum, average, count, etc.

- ***Quality Estimates for Learning Causes:*** How do we ascertain belief of learned causal sequences? Belief, in this context, is a measure of the probability with which the learned sequence truly represents a cause for the consequent. Over dense time, a counting argument fails to provide a useful measure of belief. The reason is that it is unclear how any measure that depends on counting to measure the activity of the signal would be expressed. Furthermore, the causes learned are strongly correlated with the data, and how much of the behaviour spectrum is represented by the data. When data represents observation over a small subset of behaviour, biases in the the data may also be described as part of the discovered causal sequences.

  The qualities of a good belief measure:

  - Expresses belief that a sequence is truly the cause of the consequent.
  - Is independent of biases in the data.
  - Represents bias in the data. *This maybe contradictory to the earlier point. It may also be one component of a two-component measure.*

- ***Addressing bias in data [SOLVED]:*** How would one address bias in the data, and direct learning to pick predicates which ignoring any bias that may be introduced, attributed to the way the data was sampled.

- ***Choosing an n and k:*** A time-window of W time units can be expressed using infinite combinations of $n \in \mathbb{N}$ and $k \in \mathbb{R}^{>0}$. It is important to understand the impact of n and k on different types of learning problems (over different data-sets, and different relationships).

- ***Counterfactual Reasoning:*** It is clear that associative reasoning, as done in the present work, is not as powerful as counterfactual reasoning. How can counterfactual arguments play a role in helping understand and explain events. For instance if $A \Rightarrow B$ is an association learned, the question that then arises is, *"What if A hadn't happened?"*, that is what is the C in $\neg A \Rightarrow C$?

  Alternately, let's say that at some point the relation $A \Rightarrow B$ is learned. We then assess scenario $\neg A$ and determine under what circumstances does $\neg A \Rightarrow B$. Can we use these to refine A to A' such that $A' \Rightarrow B$, but $\neg A' \not\Rightarrow B$.

- ***Incremental Learning:*** Knowledge about the world, in effect, is always evolving. The model over which learning is performed is log based, that is we loosely use the term *model* to refer to the time-series that represent a model of the observed world. How do we incrementally deal with evolving knowledge about the world?

4.2 Engineering Problems

The following is a set of engineering tasks that require design or code development, and do not involve much research. Engineering tasks are partitioned into major and minor tasks.

4.2.1 Major Tasks

- ***Graphical Interface Design for PSIMiner:*** This task involves designing and implementing a graphical interface for PSIMiner.

  Such an interface would support the following actions:

  - Specification of signals: Specify what signals are part of the learning problem, with aliases for each signal component, and its position in the time-series.
  - Importing time-series. Validation of time-series given the specification of signals.
  - Visualization of the decision tree.
  - Visualization of the learned sequences. This would include a breakdown of predicates, visualization of forward influences, matches and coverage information.

4.2.2 Minor Tasks

Minor tasks are tasks that do not involve much thought effort.

- ***Code refactoring:***
  - Elimination of dead code.
  - Elimination of internal development code.
  - Uniform indentation.
  - Documentation comments for all functions.
  - Consistency between function declarations in header files and function definitions.
  - Break large functions into smaller, fundamental operations.
- ***Debug Error Levels:*** A structured form for displaying debug information while using PSIMiner.
- ***Descriptive compiler syntax errors:*** For the configuration API, there is presently only a fundamental indicator of where the error is in the config. However, it would be useful to also have more descriptive errors.