Planning for Data Mining Tool (PDM)
Extended Abstract

Javier Ortiz and **Rubén Suárez** and **Tomás de la Rosa**
Susana Fernández and **Fernando Fernández** and **Daniel Borrajo**
Departamento de Informática
Universidad Carlos III de Madrid,
Avda. de la Universidad 30
28911 Leganés (Madrid), Spain

David Manzano
Ericsson Research Spain
Madrid, Spain

Introduction

We present a tool, PDM (Planning for Data Mining), based on Automated Planning that helps users (non necessarily experts on data mining) to perform DM (Data Mining) tasks. The starting point is a definition of the DM task to be carried out and the output is a set of plans that are executed in a DM tool to obtain a set of models and statistics. Plans are data-mining knowledge flows, i.e. sequences of DM actions that should be executed over the initial datasets to obtain the final models. However, the number of feasible plans that solve the same DM task is huge making necessary to rank them by some criterion. In a first approach, the ranking is performed following some expert estimations on the desired mining-results of the DM actions. Afterwards, these estimations are improved using machine learning techniques. In order to define the DM task, we use emerging standards, such as PMML (Predictive Model Markup Language). PMML is the leading standard for statistical and DM models and supported by over 20 vendors and organizations. With PMML, it is straightforward to develop a model on one system using one application and deploy the model on another system using another application. The PMML file is automatically translated into a planning problem described in PDDL2.1. So, any state-of-the art planner can be used to generate a plan (or plans), i.e. the sequence of DM actions that should be executed over the initial dataset to obtain the final model. Each plan or knowledge flow is executed by a machine learning engine. In our case, we employ one of the most used DM tools, WEKA (Witten and Frank 2005). In WEKA, knowledge flows are described as files with a specific format, KFML, and datasets are described as ARFF (Attribute-Relation File Format) files. The results of the DM process can be evaluated, and new plans may be requested to the planning system.

Background

This section describes two of the three languages used in this work and the files used in the learning component. First, we describe PMML (Predictive Model Markup Language), an XML based language for DM. Then, we describe KFML (Knowledge Flow for Machine Learning), another XML based language to represent DM knowledge flows for the WEKA tool (Witten and Frank 2005). The third language used in PDM, PDDL (Planning Domain Definition Language), is well known in the planning community. Finally we describe the type of files used in the learning process.

The Predictive Model Markup Language (PMML)

PMML is an XML-based markup language developed by the Data Mining Group (DMG) to provide a way for applications to define models related to predictive analytics and DM and to share those models between PMML-compliant applications. [1] It is composed of five main parts:

- The header contains general information about the file, like the PMML version, date, etc.

- The data dictionary defines the meta-data, or the description of the input data or learning examples.

- The transformation dictionary defines the applicable functions over the input data, like flattening, aggregation, average or normalization among many others. This knowledge defines the actions that can be applied over the data in the first step of the mining process.

- The models contain the definition of the DM models.

- The mining build task describes the configuration of the training task that will produce the model instance. This mining build task can be seen as the description of the sequence of actions executed to obtain the model. From the perspective of planning, it can be seen as a plan. This plan would include the sequence of DM actions that should be executed over the initial dataset to obtain the final model.

WEKA and the Knowledge Flow Files (KFML)

WEKA (Witten and Frank 2005) is a collection of machine learning algorithms to perform DM tasks. It includes all the software components needed in a DM process, from data loading and filtering to advanced machine learning algorithms for classification, regression, etc. It also includes many interesting functionalities, like graphical visualization of the results. WEKA offers two different usages. The first one is using directly the WEKA API in Java. The second

---

[1]See www.dmg.org for further information.

one consists of using the graphical tools offered. The tool included in WEKA we use is the Knowledge Flow. WEKA Knowledge Flow is a data-flow inspired interface to WEKA components. It allows to build a knowledge flow for processing and analyzing data. Such knowledge flow can include most of WEKA functionalities: load data, prepare data for cross-validation evaluation, apply filters, apply learning algorithms, show the results graphically or in a text window, etc. Knowledge flows are stored in KFML files, that can be given also as input to WEKA.

A KFML file is an XML file including two sections. The first one defines all the components involved in the knowledge flow, as data file loaders, filters, learning algorithms, or evaluators. The second one enumerates the links among the components, i.e. it defines how the data flows in the DM process, or how to connect the output of a component with the input of other components. WEKA Knowledge Flow allows loading, graphically editing, executing and saving KFML files. A knowledge flow can be seen as the sequence of steps that must be performed to execute a DM process.

Attribute-Relation File Format (ARFF)

An ARFF (Attribute-Relation File Format) file is an ASCII text file that describes a list of instances sharing a set of attributes. ARFF files were developed for use with the WEKA machine learning software.[2] ARFF files have two distinct sections. The first section is the Header information, which is followed by the Data information. The Header of the ARFF file contains the name of the relation (name of the DM task), a list of attributes (potentially including the class), and their types. The ARFF Data section of the file contains the actual instances, described in terms of the values of the attributes.

The Planning for Data Mining (PDM) Tool

Figure 1 shows the PDM architecture of the implemented system. There are four main modules; each one can be hosted in a different computer connected through a network: *Client*, *Control*, *Datamining* and *Planner*. We have used the Java RMI (Remote Method Invocation) technology that enables communication between different servers running JVM's (Java Virtual Machine). The planner incorporated in the architecture is SAYPHI (De la Rosa, García-Olaya, and Borrajo 2007) and the DM Tool is WEKA (Witten and Frank 2005). However, given that we are using standard languages other planners and/or DM tools could have been used.

The *Client* module offers an interface that provides access to all the application functionalities. It generates a PMML file from a high level description of the DM task specified by the user using the interface. Then, it sends the PMML description to the *Control* module.

The *Control* module interconnects all modules and performs the required translations. The translations needed are: from PMML to PDDL, `PMML2PDDL`; and from a PDDL plan to KFML, `Plan2KFML`. The input to the module is the DM task together with the dataset. First, the `PMML2PDDL`

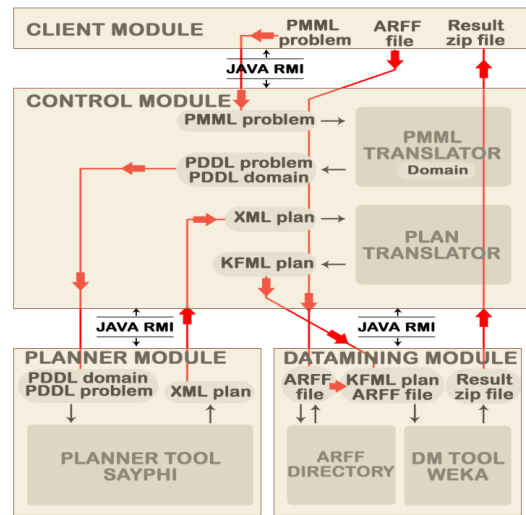[2]See www.cs.waikato.ac.nz/ ml/weka/arff.html for further information.

Figure 1: Overview of the PDM architecture.

translator generates the PDDL problem file from the PMML file. Then, the planner is executed to solve the translated problem. The returned set of plans is translated to several KFML files. Finally, the DM Tool is executed to process and run the translated KFML files. The *result* is a compressed file containing a set of directories, one for each plan. Each directory contains the model generated by the DM Tool, the statistics related to the evaluation of the model, the plans generated by the planner, and the corresponding DM workflow in KFML. This results are used to create an ARFF file. Such ARFF file let the system learn the estimations given by experts.

The *Datamining* module permits the execution of DM tasks in the WEKA DM Tool through Knowledge Flow plans. It can obtain the model output and the statistics generated as a result of the Knowledge Flow execution. This module also contains an *ARFF* directory for managing the storage of the datasets that are necessary for the WEKA executions. The input to the module is a KFML file and the output is a compressed file. The output is the model generated by the plan and the statistics related to the evaluation of the model.

The *Planning* module receives each problem and domain in PDDL format. It returns a set of plans in XML format ready for the conversion to a KFML format. Currently, planning tasks are solved by the SAYPHI planner (De la Rosa, García-Olaya, and Borrajo 2007), but the architecture could use any other planner that supports fluents, conditional effects and metrics. We have used SAYPHI because it: i) supports an extensive subset of PDDL; and ii) incorporates several search algorithms able to deal with quality metrics.

Building the models from DM Tasks in PDDL

The main challenge of our approach is how to model DM tasks by means of Automated Planning (AP). As we said, an AP task is defined by two files, the domain definition and

the problem description, whereas the DM task is defined by the PMML file.

The PMML file description

The header of the PMML file contains the following information:

- The DM goal. It can be classification, regression or clustering.

- The dataset location and size.

- The hard and soft constraints of the user. An example of hard constraint is obtaining an error lower than a given threshold, whereas minimizing the total execution time is an example of soft constraint or preference. We handle preferences and hard constraints over: i) *exec-time*, for minimizing/constraining the execution time, ii) *percentage-incorrect*, for minimizing/constraining the classification error; iii) *mean-absolute-error*, for minimizing/constraining the mean absolute error in regression tasks and clustering, and iv) *unreadability*, for maximizing the understandability of the learned model (we transform maximizing the understandability of the learned model for minimizing/constraining *unreadability*).

The data dictionary includes one field for each attribute in the dataset. The transformation dictionary includes one function for each possible filter the user wants to apply over the input data. Finally, the model part contains the definition of the DM models. The user can specify the same model but with different parameters. In general, for most DM tasks, a user would include in the PMML file the full set of WEKA DM techniques and some common settings for their parameters.

The PDDL Domain description

The PDDL domain file contains the description of all the possible DM tasks (transformations, training, test, visualization, . . .). Each DM task is represented as a domain action. The PDDL problem files contain information for a specific dataset (i.e. dataset schema, the suitable transformation for the dataset, the planning goals, the user-defined metric, etc.). Domain predicates define the state space containing static information (i.e. possible transformations, available training or evaluation tasks, etc.) and dynamic information that changes during the execution of all DM tasks (e.g. adding the fact that the dataset has already been pre-processed or evaluated). The functions allow us to define thresholds for different kinds of DM features (e.g. error, execution time threshold, or understandability of a model) and to store the values updated during the execution (e.g. total estimated error, execution time, or understandability).

Compilation of a PMML into a PDDL Problem

The `PMML2PDDL` translator automatically converts parts of a PMML file with the DM task information into a PDDL problem file. The translator uses expert knowledge to define some important planning information like the execution time of the DM tasks, the classification error for classification models or the mean absolute error in regression tasks

and clustering and the understandability of the each model. The problem file together with a domain file, that is fixed for all the DM tasks, are the inputs to the planner. The problem file contains the particular data for each DM episode, including the dataset description, the transformations and models available for that problem, and the possible preferences and constraints of the user.

Planning for DM Tasks

SAYPHI solves the planning task depending on the metric specified in the PMML file. SAYPHI includes a collection of search algorithms and domain-independent heuristics. Here, we use Best-first Search with the relaxed planning graph heuristic of FF (Hoffmann and Nebel 2001). Given that the heuristic is not admissible, it does not guarantee to find the best solution. Also, it is an open problem to assign the exact cost estimations (in terms of accuracy, time to learn, or understandability) to planning (DM) actions. So, once it finds a solution, it continues exploring nodes in order to find multiple solutions. Probably, the best models according to the planner are not necessarily the best models according to the user due to the estimation of the action cost explained below. Therefore, diversity is the only way to avoid this problem.

The Planner module outputs all the generated plans encoded in an XML file. `Plan2KFML` translates each plan into a KFML file, so it can be executed by the WEKA Knowledge Flow. The translator generates as output a new KFML file with an equivalent plan plus some extra actions. Each action in the PDDL domain corresponds to one or many WEKA components. Therefore, the translator writes for each action in the plan the corresponding set of XML tags that represent the WEKA component. Finally, the translator adds some extra components in order to save the information generated during the execution. That information is composed of the learned models and statistical information as the execution time, accuracy, . . .

Learning

As defined above, the PDM architecture uses expert knowledge to define some important planning information, like the time required to build an specific model, or the estimated accuracy of the resulting model. Initially, these values are defined by an expert. However, those estimations can be far from correct values, since they are hard to define. Also, it can become difficult to provide those values under all possible uses of the techniques and the different domains.

The goal of the learning component is to automatically acquire all those estimations from the experience of previous DM processes. The data flow of this component is described in Figure 2.

The main steps of this flow are:

1. Gathering DM Results: the goal is to gather DM experience from previous DM processes. All the information is stored in an *ARFF* file. For a given DM process, the following information is stored:

   - Information about the data set: number of instances of the data set, number of attributes of the data set, number of continuous attributes, etc.
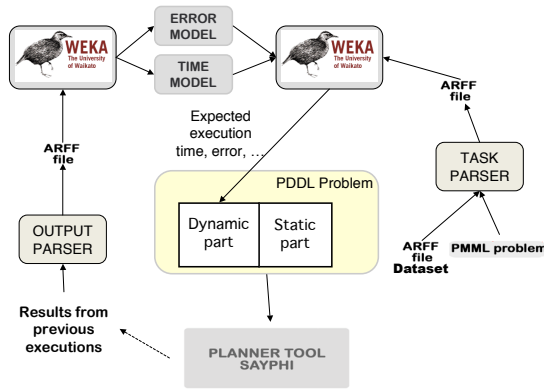
Figure 2: Learning flow in the PDM architecture.

- Information about the model to build: the type of model (association, clustering, general regression, neural network, tree, etc.), the algorithm used to learn the model (RBFNetwork, J48, etc.), the type of function (classification, regression, clustering), the learning parameters, etc.

- Information about the results obtained: type of evaluation (split, cross validation, training set), time to build the model, accuracy, mean squared error, etc.

- The plan that represents the DM workflow, and that has been executed to obtain the model

2. Model generation: the information obtained in the previous step is used to learn prediction models. The functions to learn are time, accuracy and SME (in Figure 2, error and time models. These models can be generated with the WEKA tool, as shown in the figure.

3. Given a new data set, a model, a function, and a learning algorithm, and using the models generated in the previous step, we obtain a prediction of the learning time, accuracy and SME that will be obtained if we perform a new DM process. These estimations are included in the PDDL file, so they are used when planning new DM processes. Figure 3 shows an example of how the fluents of the dynamic part of the PDDL problem file are updated. In the figure, the exec-time of a tree model and the support vector machine model are updated, among others.

There are two ways to update the PDDL problem file with these estimations: off-line and on-line. Off-line updates require to obtain information of many DM processes, use the execution information to build the models, and employ these models to update the PDDL problem file, which will stay fixed in the future. On-line updates assume that, while new data-mining processes are executed, new learning examples are obtained, so the models can be dynamically updated, and the PMML problem file is continuously updated.
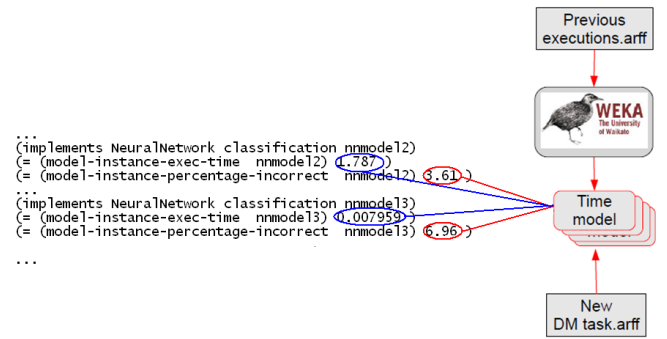

Figure 3: Learning example in the PDM architecture.

References

De la Rosa, T.; García-Olaya, A.; and Borrajo, D. 2007. Using cases utility for heuristic planning improvement. In *Case-Based Reasoning Research and Development: Proceedings of the 7th International Conference on Case-Based Reasoning*, 137–148. Belfast, Northern Ireland, UK: Springer Verlag.

Hoffmann, J., and Nebel, B. 2001. The ff planning system: Fast plan generation through heuristic search. *Journal of Artificial Intelligence Research* 14:253–302.

Witten, I. H., and Frank, E. 2005. *Data Mining: Practical Machine Learning Tools and Techniques*. 2nd Edition, Morgan Kaufmann.