Mining Actionable Patterns

P. Swapna Raj and **Balaraman Ravindran**
Department of Computer Science and Engineering
Indian Institute of Technology Madras
{pswapna,ravi}@cse.iitm.ac.in

Abstract

We propose a generic framework that uses utility in decision making to drive the data mining process. We use concepts from meta-learning and build on earlier work by Elovici and Braha, that uses decision theory for formulating an utility measure, to specialize the framework for classification tasks. We show empirical validation of the approach on a simple test domain.

Introduction

The primary motivation for the field of data mining is to provide support for decision making by detecting useful patterns in large volumes of data. The decisions that are made based on the mined patterns are often crucial to the functioning of an organization or enterprise. We call such patterns that support decision making as *actionable patterns*.

Most data mining algorithms and tools stop with mining and delivery of patterns satisfying intrinsic measures (such as accuracy, support) and ignore decision making with respect to the pattern. While researchers have looked at measuring the utility of discovered patterns in decision making, there has not been much work on using such utility measures for driving the mining process as such.

In our work we propose a framework for "closing the loop", i.e., using utility in decision making to drive the mining process. We introduce extrinsic measure which evaluate patterns with respect to the decisions made. Our goal is thus to include *decision making* as a part of Knowledge Discovery in Databases (KDD) process and come up with a general architecture to measure pattern's usefulness with respect to the decision made.

Framework

In order to present our framework for mining actionable patterns we look at mining and evaluation stages of KDD process. Data mining stage tries to optimize some intrinsic measure such as accuracy, while pattern evaluation evaluates pattern for its actionabilty based on some decision theoretic framework.

The most natural way to develop an architecture for mining actionable pattern is to merge two stages into one. To achieve this we need to develop a objective function that can drive mining of patterns based on their utility in data mining. Except in some cases this is not easy to achieve. In most formulations of the evaluation stage the optimal decision is related to patterns mined via a complex optimization procedure and no simple relationship exists to underlying intrinsic measures employed in data mining stage.

Statistical machine learning algorithms used in data mining consist of two stages : model selection and parameter estimation. While model selection is typically done manually, many parameter estimation procedures operate by gradient descent on an error/objective/likelihood function. But given the complexity of the optimization problem in the decision stage it is not usually possible to define a differentiable objective function for the data mining stage to directly optimize. Thus it is not tractable to drive the parameter estimation stage to produce actionable patterns within the existing class of data mining algorithms. Our approach is to drive the model selection stage using the utility of patterns in decision making. For this we use ideas from the field of Meta-learning.

The field of meta-learning, or learning to learn refers to a class of algorithms, that choose approriate classes of models and hyper-parameters, in order to optimize some objective function. This function is typically an intrinsic one, often the same objective function as in the parameter estimation phase. Meta-learning allows us to adjust the bias of the mining algorithm. This in turn affects the performance of the algorithm, since the patterns the algorithm detects is heavily influenced by the bias of the algorithm. In our framework we look at the utility in decision making as the performance measure which is optimized using the meta-learning approach.

We define the space of hyper-parameters, $\mathcal{S}$ that we will modify using meta-learning, say, the number of neurons in the hidden layer, and the learning rate. We then choose a neighborhood structure N, with $N(S), S \in \mathcal{S}$, denoting the neighbors of a state S. The particular neighborhood structure that we adopt is a manhattan structure, in that we allow only one hyper-parameter to change at a time, by a fixed step size. We use the *best-improvement* strategy to search the neighborhood. In order to speed up the process when the neighborhood is large, we can employ a *first-improvement* strategy, where in the first neighbor, in some arbitrary or-

	Utility 1		Utility 2		Utility 3	
	Spam	Non-spam	Spam	Non-spam	Spam	Non-spam
Accept mail	-1000	300	-500	300	-500	300
Reject mail	400	-500	400	-1000	400	-500

Table 1: Utility Matrix

Utility 1			Utility 2		Utility 3	
Experiment 1			Experiment 1		Experiment 1	
	Best utility	Avg utility	Best utility	Avg utility	Best utility	Avg utility
Before	263.30	204.53	264.85	181.49	273.15	269.31
After	266.19	264.02	266.76	234.26	275.10	272.97
Experiment 2			Experiment 2		Experiment 2	
Before	266.42	214.01	265.82	217.32	276.06	241.142
After	266.42	262.87	265.82	245.72	276.77	273.66
Experiment 3			Experiment 3		Experiment 3	
Before	263.43	259.85	245.47	236.58	275.03	260.39
After	269.09	266.34	245.47	242.64	275.03	261.82

Table 2: Best and Average utility

der, with a better evaluation than the current state is chosen. Since we only find local optima, we repeat the entire local search procedure several times by picking different start states for the hyper-parameters. We finally use the hyper-parameter setting that gave us the best evaluation across all repetitions.

	Utility 1	Utility 2	Utility 3
Experiment 1	60.05	126.06	112.05
Experiment 2	127.94	100.08	150.53
Experiment 3	56.62	54.37	85.74

Table 3: Average Performance

Experiments

In order to empirically validate our architecture, we chose a Spambase dataset from the UCI Machine Learning Repository which consists of 4601 instances (1813 Spam = 39.4 %) and 58 attributes (57 continuous, 1 nominal class label). The class labels are spam(1) or non-spam(0). The actions that can be taken by the decision-maker are either "accept mail" or "reject mail". The consequence of rejecting a spam mail or accepting non-spam mail carries a certain reputation cost that can be quantified.

The mining approach we considered is a feed forward neural network (FFNN). We made this choice, since the FFNN offered a plethora of hyper-parameters to tune. The hyper-parameters used in experiment 1 : learning rate, number of hidden neurons and transfer function while in experiment 2 : number of hidden neurons and transfer function and in experiment 3 : number of hidden neurons and learning algorithm. The range of the learning rate was between 0.01 and 0.8, with the step size of 0.01. The number of hidden neurons could vary between 1 and 15, with a step size of 1. The transfer function could be either a hyperbolic sigmoid or a logarithmic sigmoid. The learning algorithms considered were gradient descent without momentum or with a fixed momentum term.

We employed the decision theoretic framework from (Elovici and Braha 2003) to evaluate our trained classifiers. We study the effect of different base utility functions (Table 1) on our architecture and the results are tabulated in Table 2. All numbers are averages over 10 meta-learning trials. Note that the higher the utility the greater the "actionability" of the patterns mined. We compute the best and average utilities achieved before the local search and after the local search procedure. The best "before" performance quantifies the best among the 10 random hyper-parameter settings. The best "after" performance quantifies the best with the meta-learning architecture. It is not always the case that we achieve the best after performance by starting at the best before hyper-parameter state.

Average before and after performance for the different experiments are also given in Table 2. These numbers tells us that the average utility expected with the tuned hyper-parameters is a lot higher than the randomly chosen ones. Thus we would need a lot fewer learning trials to get to more actionable patterns. The average gain in performance for various utility functions are tabulated in Table 3.

In this framework we have used utility function to model the trade-off in the decision making process. It is difficult for the user to quantify the trade-off accurately. Hence one may use less informative formulation such a reward function as employed in (SwapnaRaj and Ravindran 2008).

References

Elovici, Y., and Braha, D. 2003. A decision-theoretic approach to data mining. *IEEE Transactions on Systems, Man, and Cybernetics, Part A* 33(1):42–51.

SwapnaRaj, P., and Ravindran, B. 2008. Personalized webpage rendering system. In *14th International Conference on Management of Data*.