Full Model Selection in the Space of Data Mining Operators

Quan Sun, Bernhard Pfahringer and Michael Mayo
Department of Computer Science
The University of Waikato, New Zealand
{qs12, bernhard, mmayo}@cs.waikato.ac.nz

1 Introduction

In this paper, we propose a framework and a novel algorithm for the full model selection (FMS) problem. The proposed algorithm, combining both genetic algorithms (GA) and particle swarm optimization (PSO), is named GPS, in which a GA is used for searching the optimal structure of a data mining solution, and PSO is used for searching the optimal parameter set for a particular structure instance.

2 The DMO space

We define a search space that consists of all data mining operators that are applicable to a given dataset for a user-specified goal, such as a set of outlier filters, a set of feature selection methods, a set of data transformation techniques and a set of machine learning algorithms. In this sense, we call the subject of interest "the space of data mining operators (DMO)", or simply "the DMO space".

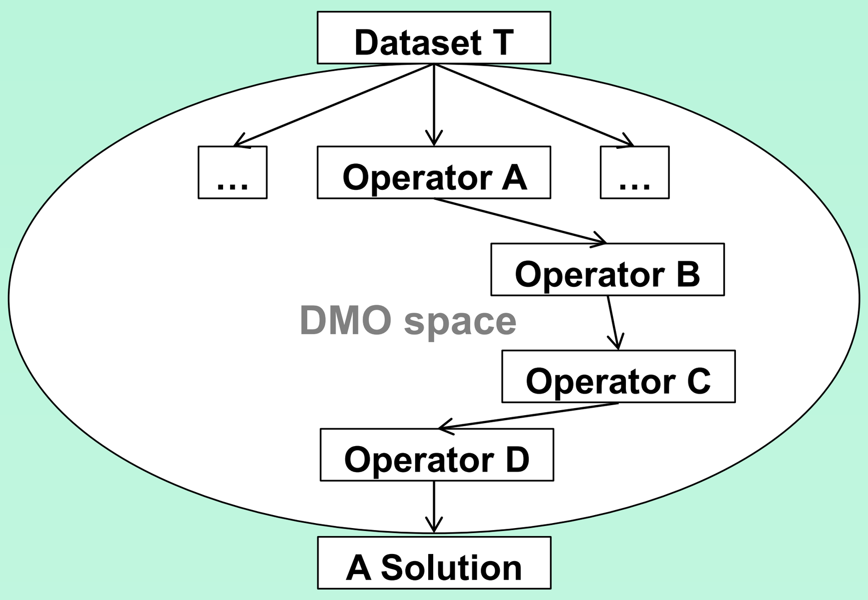

Figure 1. An illustration of the DMO space.

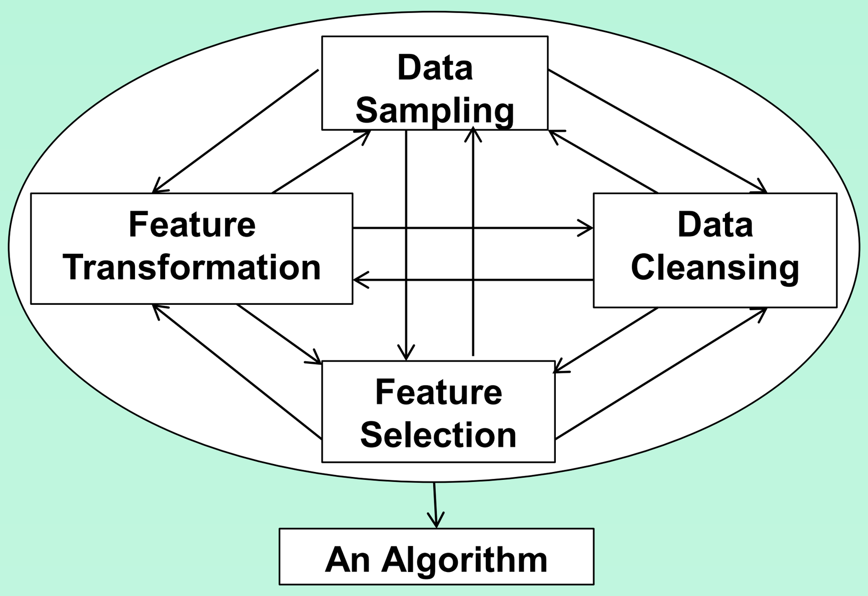

Figure 2. A graphical representation of the DMO template.

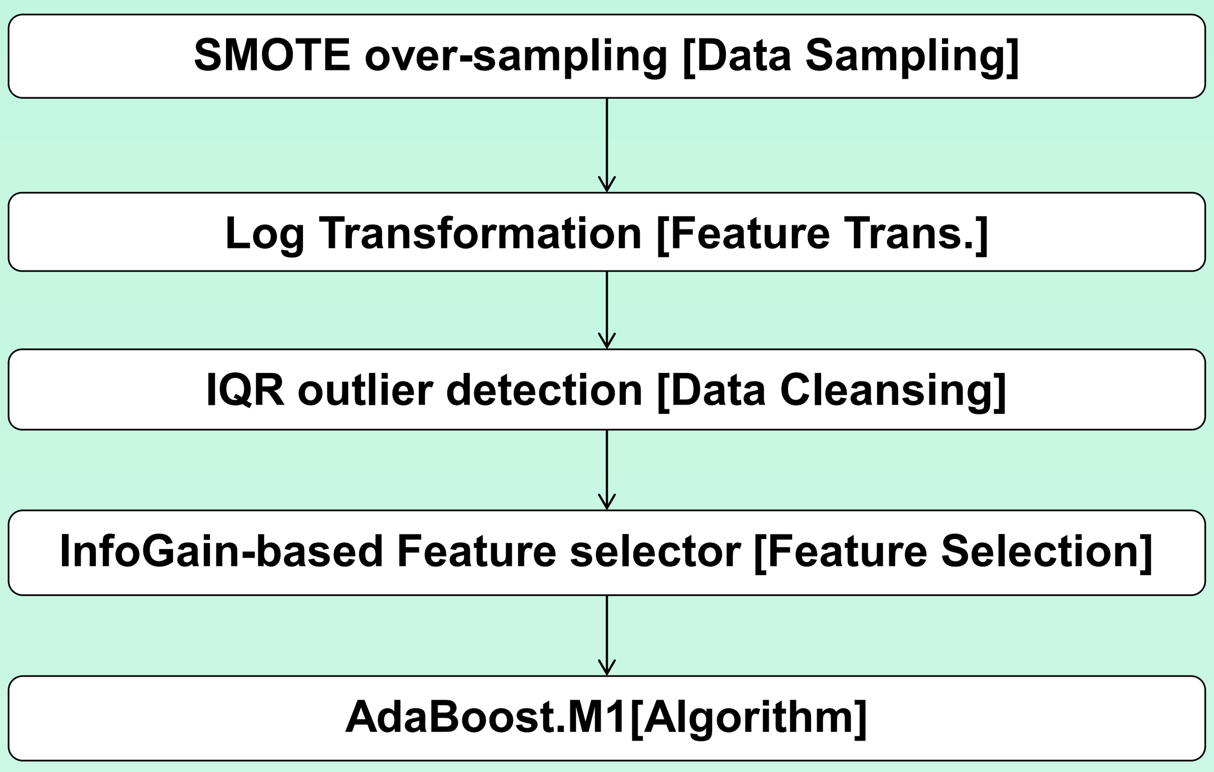

Figure 3. A graphical representation of a DMO solution template instance.

Data Sampling	Date Cleansing	Feature Trans.	Feature Sel.
SMOTE oversampling Resample with replacement Resample without replacement Do nothing	NumericCleaner RemoveUseless ReplaceMissingValues Do nothing	Normalize Standarlize Center AddNoise Discretize NominalToBinary NumericTransformation Do nothing	CfsSubsetEval InfoGainAttributeEval GainRatioAttributeEval PrincipalComponents ChiSquaredAttributeEval Do nothing

Algorithm	HyperParameters
Bagging with RandomTree	Num.Bagging.Iterations[int], Num.Atts[int], Tree.Depth[int]
Bagging with REPTree	Num.Bagging.Iterations[int], Num.Folds[int], Tree.Depth[int]
AdaBoost.M1 with DecisionStump	Num.Boosting.Iterations[int], UseResample[boolean]
LogitBoost with DecisionStump	Num.Boosting.Iterations[int], UseResample[boolean]
Bagging with J48 Decision Tree	Num.Bagging.Iterations[int], TreePrune[boolean], Conf.[real]
RotationForest with REPTree	Num.Iterations[int], PctRemoved [real], Projection {PCA, RandomProj}

Table 1. WEKA [2] algorithms and filters that are used as the DMO objects in the GPS algorithm

3 The GA-PSO-FMS (GPS) system

The basic steps of the GPS algorithm are: for each GA iteration, firstly a population of DMO template instances is randomly generated (Figure 2 and Figure 3). Then, the placeholders of each template instance are randomly populated with the DMO objects in Table 1. Then, PSO is used for searching an optimal parameter setting for each template instance. The population is sorted by PSO-based evaluation scores. At the end of each GA iteration, typical GA operators can be applied for generating new template instances. The above procedure is repeated ***T*** times.

```
BasicGPSProcedure(T,P,M,W,G):
Input: T (num.generations for GA), P (population size for GA),
M (num.evolutions for PSO), W (swarm size for PSO), G (goal
metric)

▶
for i ← 1 to T
    Get P random template instances
    Populate template instances with objects in Table 1
    Use BasicPSOProcedure(M,W,G,I) to search for the
    optimal parameter set for each template instance I
    (optimizing the goal metric G), and assign an evaluation
    score to each template instance I
    Sort the population by evaluation scores
    Do crossover // we use simple one point crossover
    between the top 20% template instances
    Do mutation // we randomly choose 30% template
    instances from the population, and randomly change one
    DMO in each template instance
    Replace the last N template instances with the N new
    template instances generated in crossover and mutation,
    where N = (0.2 + 0.3 ) x P
    solution-best ← population-best
endfor
return solution-best
```

```
BasicPSOProcedure(M,W,G,I,c1=2.0,c2=2.0):
Input: M (num.evolutions), W (swarm size), G (goal
metric, default AUC), I (a solution template instance),
c1, c2 are the weighting coefficients for the personal
best the global best positions, default c1=c2=2.0

▶
population ← Ø
particle p.global-best ← Ø
Initialize swarm (W particles) based on I
for i ← M
    foreach particle p in population do
        UpdateVelocity(p, c1, c2)
        UpdatePosition(p)
        p.G ← EvaluateTemplateInstance(p,G,I)
        if p.G ≥ p.personal-best
            p.personal-best ← p.position
            if p.G ≥ p.global-best
                p.global-best ← p.personal-best
    endfor
endfor
return p.global-best
```

Figure 4. Pseudocode of the GPS algorithm

4 Experimental Results

We experiment with ten real-world classification problems. To test the performance of the GPS algorithm, we implemented a variant of the PSMS system (a PSO-based FMS algorithm) proposed in [1] with the DMO objects defined in Table 1. Figure 6 shows a summary of a comparison of AUC performance between GPS and PSMS under 30 different configurations over ten datasets.

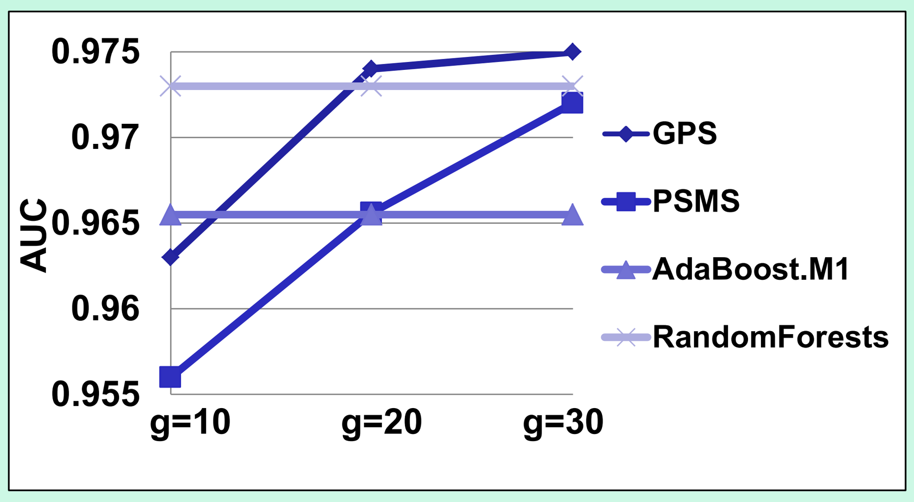

Figure 5. AUC performance comparison for the MiniBooNe. particle identification dataset.

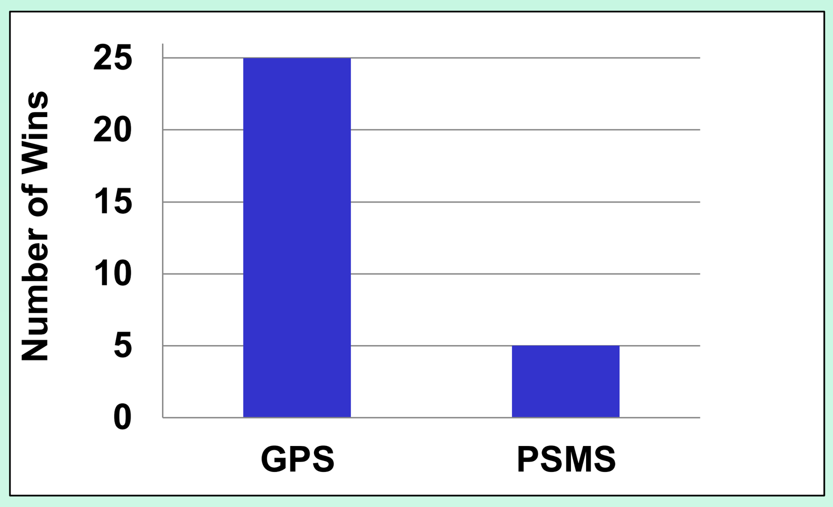

Figure 6. AUC performance between GPS and PSMS under 30 different configurations.

5 Conclusions

Our experimental results show that the GPS algorithm outperforms the PSMS system, the state-of-the-art PSO-based FMS algorithm on the ten real-world datasets studied in this paper. In the longer version of this paper, we also theoretically examined the feasibility of using the divide and conquer idea for speeding up the GPS algorithm.

References

[1] H. J. Escalante, M. Montes, and L. E. Sucar. Particle swarm model selection. *Journal of Machine Learning Research*, 10:405 – 440, 2009.

[2] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. Witten. The WEKA data mining software: An update. *SIGKDD Explorations*, 11(1), 2009..