A APPENDIX

A.1 PROPERTIES OF THE TEST INSTANCES

We have used 22 datasets in our experiments. The properties of these datasets are summarized in Table 3. The first 17 datasets are for binary classification, whereas the remaining datasets are for multi-class classification.

Table 3: The properties of the datasets. Here, *SAMPLE SIZE*, *CLASSES* and *FEATURES* stand for the number of observations, the number of classes and the number of features, respectively. The datasets are from the following sources: OILSPILL from Kubat et al. (1998), PHONEME from an online repository, remaining from Dua & Graff (2017).

INSTANCE	SAMPLE SIZE	CLASSES	FEATURES
BANKNOTE	1372	2	4
HEARTS	303	2	13
ILPD	583	2	10
IONOSPHERE	351	2	34
LIVER	345	2	6
PIMA	768	2	8
TIC-TAC-TOE	958	2	9
TRANSFUSION	748	2	4
WDBC	569	2	31
ADULT	32561	2	14
BANK-MKT	11162	2	16
MAGIC	19020	2	10
MUSHROOM	8124	2	22
MUSK	6598	2	166
OILSPILL	937	2	49
PHONEME	5404	2	5
MAMMOGRAPHY	11183	2	6
SEEDS	210	3	7
WINE	178	3	13
GLASS	214	6	9
ECOLI	336	8	7
SENSORLESS	58509	11	48

A.2 PARAMETRIC MODEL

Before we analyze the parametric model, let us first rewrite problem equation 3 in *canonical form* using matrix notation. To simplify our exposition, we define

$$\boldsymbol{z} = \begin{bmatrix} \boldsymbol{w} \\ \boldsymbol{v} \\ \boldsymbol{u} \end{bmatrix}, \ \boldsymbol{d} = \begin{bmatrix} \boldsymbol{c} \\ \boldsymbol{e} \\ \boldsymbol{0} \end{bmatrix}, \ \boldsymbol{b} = \begin{bmatrix} \boldsymbol{e} \\ \boldsymbol{e} \end{bmatrix} \text{ and } \boldsymbol{M} = \begin{bmatrix} \hat{\boldsymbol{A}} & \boldsymbol{I} & \boldsymbol{0} \\ \boldsymbol{A} & \boldsymbol{0} & \boldsymbol{I} \end{bmatrix},$$

where $\boldsymbol{w} = [w_j]_{j\in\mathcal{J}}$, $\boldsymbol{v} = [v_i]_{i\in\mathcal{I}}$, $\boldsymbol{c} = [c_j]_{j\in\mathcal{J}}$ $\hat{\boldsymbol{A}} = [\hat{a}_{ij}]_{i\in\mathcal{I},j\in\mathcal{J}}$, $\boldsymbol{A} = [a_{ij}]_{i\in\mathcal{I},j\in\mathcal{J}}$ along with $\boldsymbol{e}$ and $\boldsymbol{0}$ denoting vector of ones and zeros, respectively. The auxiliary vector $\boldsymbol{u}$ is used as a slack variable to obtain a model in canonical form. With this notation, the master problem and its dual become

$$\min\{\boldsymbol{d}^\intercal \boldsymbol{z} : \boldsymbol{M}\boldsymbol{z} = \boldsymbol{b}, \boldsymbol{z} \geq \boldsymbol{0}\} \tag{7}$$

and

$$\max\{\boldsymbol{b}^\intercal \boldsymbol{\alpha} : \boldsymbol{M}^\intercal \boldsymbol{\alpha} \leq \boldsymbol{d}\}, \tag{8}$$

respectively. We note that the first set of constraints in equation 3 is also written as an equality. This is still a valid formulation, since the vector $\boldsymbol{v}$ plays the role of slack variables. Recall that the second set of constraints in the master problem guarantees coverage of the samples. Without loss of generality, we have set the right-hand-side of these constraints to one, *i.e.* $\varepsilon = 1$. Therefore, one may define a hyperparameter $\delta > -1$ and consider a set of alternative right-hand-side values $\boldsymbol{b}(\delta) = \boldsymbol{b} + \delta\bar{\boldsymbol{b}}$ with $\bar{\boldsymbol{b}}^\intercal = [\boldsymbol{0}^\intercal, \boldsymbol{e}^\intercal]$. Such a change in the right-hand-side value leads to a parametric linear program. Next, we give an interval of δ values so that as long as δ remains in this interval, the optimal basis does not change and there is no need to solve the master problem again.

Suppose that we have solved problem equation 7 and obtained the optimal solutions $\boldsymbol{z}^*$ and $\boldsymbol{\alpha}^*$. Let us denote the indices of the basic and nonbasic variables with $\mathcal{B}$ and $\mathcal{N}$, respectively. By rearranging the corresponding columns, we can write

$$\boldsymbol{M}\boldsymbol{z}^* = \boldsymbol{M}_{\mathcal{B}}\boldsymbol{z}^*_{\mathcal{B}} + \boldsymbol{M}_{\mathcal{N}}\boldsymbol{z}^*_{\mathcal{N}} = \boldsymbol{b}$$

and

$$\boldsymbol{d}^\intercal \boldsymbol{z}^* = \boldsymbol{d}_{\mathcal{B}}^\intercal \boldsymbol{z}_{\mathcal{B}}^* + \boldsymbol{d}_{\mathcal{N}}^\intercal \boldsymbol{z}_{\mathcal{N}}^*.$$

Hence, we obtain $\boldsymbol{z}_{\mathcal{B}}^* = \boldsymbol{M}_{\mathcal{B}}^{-1}\boldsymbol{b}$, $\boldsymbol{z}_{\mathcal{N}}^* = \boldsymbol{0}$ and $\boldsymbol{\alpha}^* = \boldsymbol{d}_{\mathcal{B}}\boldsymbol{M}_{\mathcal{B}}^{-1}$. Clearly, replacing $\boldsymbol{b}$ with $\boldsymbol{b}(\delta)$ does not affect the feasibility of the dual solution. The primal solution, however, becomes

$$\boldsymbol{z}_{\mathcal{B}}^*(\delta) = \boldsymbol{M}_{\mathcal{B}}^{-1}\boldsymbol{b}(\delta) = \boldsymbol{M}_{\mathcal{B}}^{-1}\boldsymbol{b} + \delta\boldsymbol{M}_{\mathcal{B}}^{-1}\bar{\boldsymbol{b}} = \boldsymbol{z}_{\mathcal{B}}^* + \delta\bar{\boldsymbol{z}}_{\mathcal{B}}^*.$$

We observe for $\boldsymbol{z}_{\mathcal{N}}^*(\delta) = \boldsymbol{0}$ that

$$\boldsymbol{M}_{\mathcal{B}}\boldsymbol{z}_{\mathcal{B}}^*(\delta) = \boldsymbol{b}(\delta),$$
$$\boldsymbol{d}^\intercal \boldsymbol{z}_{\mathcal{B}}^*(\delta) = \boldsymbol{b}(\delta)^\intercal \boldsymbol{\alpha}^*.$$

This shows that current basis remains optimal as long as $\boldsymbol{z}_{\mathcal{B}}^*(\delta) \geq \boldsymbol{0}$. This leads to the following simple interval for δ values:

$$\max_{i\in\mathcal{B}, b_i'>0}\{-1, -\tfrac{z_i^*}{\bar{z}_i^*}\} \leq \delta \leq \min_{i\in\mathcal{B}, b_i'<0}\{-\tfrac{z_i^*}{\bar{z}_i^*}\},$$

where $[z_i^*]_{i\in\mathcal{B}} = \boldsymbol{z}_{\mathcal{B}}^*$ and $[\bar{z}_i^*]_{i\in\mathcal{B}} = \bar{\boldsymbol{z}}_{\mathcal{B}}^*$. During hyperparameter tuning, if δ goes out of this interval, then problem equation 8 can be solved after replacing $\boldsymbol{b}$ with $\boldsymbol{b}(\delta)$. Consequently, the same line of parametric analysis can be applied to find the next interval of values.

A.3 Relation to Boosting Methods

The relation between the proposed rule generation algorithm (RUG) and the boosting methods can be summarized in three parts:

1. In boosting methods, a sequence of weak classifiers are trained in tandem, and each classifier depends on the classifiers that come before it. To improve accuracy with every new classifier, the weights of the misclassified points are increased. Moreover, each classifier in the sequence receives a weight according to its classification performance, while the weights of the previous classifiers are fixed. In our master problem equation 3, we also assign weights to the rules and these rules can be considered as weak classifiers. However, when we apply rule generation, we do not fix the weights of rules. Instead, we obtain the optimal rule weights by solving a linear program at each iteration. Therefore, rule weights are determined simultaneously rather than being assigned sequentially.

2. The duality discussion with our linear programs also lends itself to an interesting discussion about our approach and the margin maximization idea in boosting methods as presented by Schapire et al. (1998), Grove & Schuurmans (1998) and Demiriz et al. (2002). These authors establish that a sample with a large margin is likely to be classified correctly. Thus, margin maximization is about assigning larger weights to those samples with small margins. Clearly, the first set of constraints in equation 3 always holds as equalities due to the nonnegative costs of the variables v_i, $i \in \mathcal{I}$. Using complementary slackness conditions in linear programming, we have the following facts:

   - $\sum_{j\in\mathcal{J}_k} \hat{a}_{ij} w_j > 1$ implies $v_i = 0$ and $\beta_i = 0$.
   - $\beta_i > 0$ only if $\sum_{j\in\mathcal{J}_k} \hat{a}_{ij} w_j \leq 1$.
   - $0 < \beta_i < 1$ only if $\sum_{j\in\mathcal{J}_k} \hat{a}_{ij} w_j = 1$ (marginal accuracy for sample i).
   - $\sum_{j\in\mathcal{J}_k} \hat{a}_{ij} w_j < 1$ implies $\beta_i = 1$ (misclassification of sample i).

   This shows that the optimal dual variable $\beta_i^{(t)}$ becomes positive only if sample i is misclassified, or it is correctly classified but remains on the boundary. So at the next iteration of rule generation, only those samples that have small margins are considered.

3. Mason et al. (2000) relate boosting algorithms to gradient descent. It is well-known that the dual optimal solution plays the role of gradient vector for the primal problem. This points out yet another connection between our approach and the boosting methods.

Table 4: The performances of CG Dash et al. (2020), BRS Wang et al. (2017), AM and BCD Su et al. (2016), RIPPER Cohen (1995), GLRM Wei et al. (2019), RULEFIT Friedman et al. (2008) and RUG-T. Here, *ACC.*, and *COMP.* stand for accuracy and complexity, respectively. Complexity is defined as the summation of the number of generated rules and the number of conditions in the rules. The figures in columns 2-11 and 12-15 are taken from Dash et al. (2020) and Wei et al. (2019), respectively.

DATASET	CG		BRS		AM		BCD		RIPPER		GLRM		RULEFIT		RUG-T	
	ACC.	COMP.	ACC.	COMP.	ACC.	COMP.	ACC.	COMP.	ACC.	COMP.	ACC.	COMP.	ACC.	COMP.	ACC.	COMP.
BANKNOTE	0.991	25	0.991	30.4	0.985	24.2	0.987	21.3	0.992	28.6	0.999	47.7	0.999	1124.9	0.988	48.7
HEARTS	0.789	11.3	0.789	24	0.729	11.5	0.742	15.4	0.793	16	0.846	5.2	0.833	59.4	0.822	19.7
ILPD	0.696	10.9	0.698	4.4	0.715	0	0.715	0	0.698	9.5	0.708	1.9	0.717	2106.0	0.705	10.6
IONOSPHERE	0.900	12.3	0.869	12	0.909	16	0.915	14.6	0.880	14.6	0.909	150.7	0.943	1225.6	0.909	24.9
LIVER	0.597	5.2	0.536	15.1	0.557	8.7	0.519	4	0.571	5.4	0.580	34.7	0.586	89.7	0.693	58.6
PIMA	0.741	4.5	0.743	17.4	0.732	2.7	0.734	2.1	0.734	17	0.758	15.5	0.747	3211.5	0.737	30.9
TIC-TAC-TOE	1.000	32	0.999	32	0.843	24.9	0.815	12.6	0.982	32.9	0.980	67.1	1.000	1640.7	0.906	70
TRANSFUSION	0.779	5.6	0.766	6	0.762	0	0.762	0	0.789	6.8	0.793	11.9	0.747	3211.5	0.755	26.9
WDBC	0.940	13.9	0.947	16	0.958	11.6	0.958	17.3	0.930	16.8	0.982	228.4	0.968	562.3	0.937	28.8
ADULT	0.835	88	0.817	39.1	0.830	15	0.824	13.2	0.836	133.3	0.859	94.2	0.870	719.9	0.851	43.8
BANK_MKT	0.900	9.9	0.874	13.2	0.900	6.8	0.897	2.1	0.899	56.4	0.901	83.6	0.887	0.2	0.820	85.6
MAGIC	0.853	93	0.825	97.2	0.807	11.5	0.803	9	0.845	177.3	0.854	196.2	0.875	1656.0	0.834	72.8
MUSHROOM	1.000	17.8	0.997	17.5	0.999	15.4	0.999	14.6	1.000	17	1.000	18.2	1.000	927.9	1.000	31
MUSK	0.956	123.9	0.933	33.9	0.969	101.3	0.921	24.4	0.959	143.4	0.984	1079.7	0.978	2000.3	0.925	87.2
AVERAGE	0.856	32.4	0.842	25.6	0.835	17.83	0.828	10.8	0.851	48.2	0.868	145.4	0.868	1324.0	0.849	45.7

A.4 COMPARISON AGAINST RECENT BINARY CLASSIFICATION METHODS FOR INTERPRETATION

Table 4 presents a comparison of RUG against existing studies from the literature. In the table, "CG" stands for the column generation framework by Dash et al. (2020), "BRS" is the Bayesian Rule Set approach by Wang et al. (2017), "AM" and "BCD" are alternating minimization and block coordinate descent algorithms by Su et al. (2016), respectively. "RIPPER"[3] is the well-known rule generation heuristic of Cohen (1995). The last two methods "GLRM" (short for Generalized Linear Rule Models) and "RULEFIT" are rule ensemble approaches by Wei et al. (2019) and Friedman et al. (2008), respectively. Both GLRM and RULEFIT[4] results are the outputs reported for numerical features in the work by Wei et al. (2019). The acronym "ACC." stands for the mean accuracy and "COMP." denotes the complexity. Dash et al. (2020) define complexity as the sum of the number of rules and the number of conditions for each rule. Although complexity is not directly addressed in our study, it is calculated for RUG for the sake of completeness. Wei et al. (2019) uses weighted number of rules implying the same complexity concept (or a lower bound on the corresponding complexity values) in their models using numerical features. To make a fair comparison, we have additionally considered RUG with threshold values (RUG-T), for which the number of subproblem calls is restricted to five, and the rules satisfying the weight threshold of $w_j \geq 0.05$, $j \in \mathcal{J}$ are selected for testing. The last two columns Table 4 give the mean accuracies and the complexity values for RUG-T, respectively. Dash et al. (2020) consider only binary classification. Thus, the multi-class classification datasets in the previous tables are excluded. Moreover, two datasets from Dash et al. (2020) are excluded since we have failed to access those instances. The average accuracy of RUG-T is on par with GLRM, RULEFIT and CG where the latter has superior accuracy than BRS, AM, BCG and RIPPER. It can be observed that rule ensemble methods GLRM and RULEFIT has superior accuracy. This comes with an expense of intepretability and their complexity is higher than other methods including RUG-T and CG. GLRM has a complexity of more than three times RUG-T has. Complexity of the RULEFIT is not very promising and yields the worst outcome indicating that its interpretability is not good. The average complexity of RUG-T is slightly worse than that of CG except MUSK dataset. RUG-T exposes a worthwhile example on how to fine-tune parameters of RUG so that a good balance can be achieved for interpretability. We also observe that RUG is robust in the sense that reducing the number of RMP calls does not reduce the accuracy drastically. Besides, applying a threshold to select rules contributes to interpretability without a severe damage on accuracy. We are unable to compare computational time requirements of GLRM and RULEFIT as it is not reported in the work by Wei et al. (2019). Fortunately, RUG-T is significantly faster than the CG. Indeed, all our algorithms run in less than a minute, while CG is reported to take around 20 minutes for all datasets except MUSHROOM and TIC-TAC-TOE.

[3]This is the JRip implementation in Weka.

[4]These two models are respectively named as LRRN and RULEFITN in the original work by Wei et al. (2019).

A.5 CASE STUDY: FURTHER DETAILS

We next compare the rules extracted by RUG against the rules (leaves) of a DT trained on the same dataset with the same setting (tree depth of three). Figure 3 lists the sets of rules resulting from RUG and DT in blocks (a) and (b), respectively. DT yields seven rules (leaves) with an accuracy of 0.92. To make a fair comparison, only the first three rules of RUG is shown where the accuracy of the RUG reaches 0.93. Rules R_1 to R_3 are in descending order of their weights. As a reminder, these are the same first three rules shown in Figure 2. Even with fewer number of rules, the accuracy of RUG is better than DT.

R_1: If Marginal Adhesion ≥ 8 **then** Malignant
R_2: If Bare Nuclei ≤ 2 **and** Uniformity of Cell Shape ≤ 3 **then** Benign
R_3: If Marginal Adhesion ≥ 2 **and** Uniformity of Cell Size ≥ 5 **then** Malignant

(a) RUG

D_1: If Uniformity of Cell Shape ≤ 3 **and** Bare Nuclei ≤ 5 **and** Bare Nuclei ≤ 2 **then** Benign
D_2: If Uniformity of Cell Shape ≤ 3 **and** Bare Nuclei ≤ 5 **and** Bare Nuclei ≥ 3 **then** Benign
D_3: If Uniformity of Cell Shape ≤ 3 **and** Bare Nuclei ≥ 6 **and** Uniformity of Cell Shape ≤ 1 **then** Benign
D_4: If Uniformity of Cell Shape ≤ 3 **and** Bare Nuclei ≥ 6 **and** Uniformity of Cell Shape ≥ 2 **then** Malignant
D_5: If Uniformity of Cell Shape ≥ 4 **and** Uniformity of Cell Size ≤ 1 **then** Benign
D_6: If Uniformity of Cell Shape ≥ 4 **and** Uniformity of Cell Size ≥ 2 **and** Bland Chromatin ≤ 2 **then** Malignant
D_7: If Uniformity of Cell Shape ≥ 4 **and** Uniformity of Cell Size ≥ 2 **and** Bland Chromatin ≥ 3 **then** Malignant

(b) DT

Figure 3: Rules generated by RUG versus the rules (leaves) obtained with DT. RUG rules are shown in descending order of their weights.

We also observe that RUG extracts more interpretable rules than DT. A drawback of DT rules (D_1-D_7) is their interdependent structure. For example D_3 is constructed as the complement of the statement "Bare Nuclei $\leq$ 5" in D_1 and D_2. However, D_3 has two observations; one originally labeled as malignant and the other as benign. This implies a false negative classification of a patient in one out of two cases (50%). Similar structure can be observed for DT rules D_5 to D_7, where they are created as the complement of "Uniformity of Cell Shape $\leq$ 3" in rules D_1 to D_4. Such an unbalanced structure is not observed for the rules extracted by RUG. More importantly, a false negative classification can be compensated in two ways by RUG: First, multiple rules covering a sample gives more chance to correct classification of the sample. In case there are two or more rules that do not agree, the rule weights can compromise the final class of a sample and correctly classify it based on the loss function's structure. An example of such a case can be illustrated using the false negative classification of the sample with D_3. Using RUG, the patient data is covered by four rules; R_4, R_9 and R_{15} with weights 0.43, 0.29 and 0.14 as malignant, and R_{13} with weight 0.14 as benign. Using the labels assigned by the rules and their weights, this sample is classified as malignant. This implies that, RUG has detected the tumor and made the correct diagnosis for the patient. Second way comes from the flexibility of the LP model behind RUG. The rules that yield false negative classifications can be explicitly penalized in the objective function. That is, each rule yielding a false negative classification of a sample can be assigned a high cost coefficient (penalty), and consequently, model equation 3 can be enforced to choose another rule. As an example, R_{13} can be penalized for this case and left outside the solution to promote correct classification.

A.6 PRELIMINARY COMPUTATIONAL EXPERIMENTS ON LARGE INSTANCES

We have performed a series of experiments to show the scalability of our approaches on large scale instances. To that end, we have used "Forest CoverType", a multi-class large size dataset from the UCI repository. The dataset has 581012 samples with 54 features and consists of seven classes. We have considered the 10 numerical features, and a subset of the samples (e.g. first 50000 samples in the dataset) for prediction. In Table 5, the first column presents the number of samples used for each instance. We have kept all the parameters the same as described for our computational experiments except maximum tree depth which is set to three in all algorithms. We observe that on average RUG has the highest accuracy with the least number of rules. Thus, RUG outperforms other algorithms in terms of both accuracy and interpretability. Surprisingly, we also observe that RUX provides superior results than both of its variants based on RF and ADA. This becomes more visible for ADA versus RUX-ADA comparison. Average training time of the algorithms are 408.08, 128.23 and 141.45 seconds for RUX-RF, RUX-ADA and RUG, respectively. Note that, since RUX includes training times of its source algorithm, e.g. RF and ADA, RUG has lower computational requirement than RF. In fact, ADA's training time steadily increases with increasing number of samples from 50k to 300k. RUG's training time is lower than ADA for instances having 200k, 250k, and 300k samples. This implies the scalability of the RUG for larger instances. Lastly, the prediction times are also much shorter for RUG. In average, RUX-RF, RUX-ADA, and RUG have 382.99, 219.76,and 28.47 seconds of prediction time, respectively. This implies that RF and ADA have more than 13 and seven times longer prediction time than RUG.

Table 5: The performances of Random Forest (RF), AdaBoost (ADA) as well as Rule Extraction applied to RF and ADA - denoted as RUX-RF and RUX-ADA -, and RUG. In the table, *SAMPLES*, *ACC.*, and *# RULES* stand for number of samples used from the "Forest CoverType" dataset from the UCI repository, accuracy and number of rules, respectively.

SAMPLES	RF		ADA		RUX-RF		RUX-ADA		RUG	
	ACC.	# RULES	ACC.	# RULES	ACC.	# RULES	ACC.	#RULES	ACC.	# RULES
50000	0.704	800	0.570	400	0.748	147	0.720	143	0.729	22
100000	0.750	800	0.656	400	0.770	172	0.760	102	0.764	12
150000	0.771	800	0.680	400	0.792	167	0.757	150	0.796	13
200000	0.739	800	0.586	400	0.749	218	0.717	85	0.764	13
250000	0.750	800	0.642	400	0.723	172	0.676	102	0.754	13
300000	0.732	800	0.688	400	0.700	189	0.707	92	0.731	11
AVERAGE	0.741	800	0.637	400	0.747	177.5	0.723	112.333	0.756	14