Online Supplement

Solving a Class of Cut-Generating Linear Programs via Machine Learning

Atefeh Rajabalizadeh Danial Davarnia

Appendix A: Omitted Computational Results

In this section, we present the detailed reports for the classification results for the problems studied in Section 4. First, we provide the numerical results for the CHM problem. In particular, Tables 4–8 report the classification results for the random forest (RF), the k-nearest neighbor (KNN), logistic regression (LR), support vector machines (SVM), and neural network (NN), respectively. For each of the classification methods, the main algorithmic choices and parameter settings are reported in the caption of the corresponding tables. These parameter values are selected through tuning and cross validation. Once selected, these settings remain the same as "universal settings" across all problem instances and size categories.

In Tables 4–8, columns "Size" and "#" represent the size category and the test instance number, respectively. Column "data" distinguishes the results for the training and the test set for each instance. The entries of the *confusion matrix* as a result of the training are given in columns "TN", "FP", "FN" and "TP". Column "Accuracy" reports the accuracy of the trained classifier calculated as $\frac{\text{TN+TP}}{\text{TN+TP+FP+FN}}$. The entry under columns "ML Time" and "CGLP Time" are defined similarly to those in Table 1 in Section 4.1.

Next, we present detailed reports for some instances of the binary programs studied in Section 4.2 that include the classification results for each layer of the B&B tree. Table 9 shows the results for the first instance, i.e., instance #1 of size $n = 40$ and $m = 50$, of the multi-knapsack problems of Table 2. Similarly, Table 10 shows the results for the first instance, i.e., instance `p0033`, of the benchmark problems of Table 3. In these tables, each row contains the classification outcome for each layer of the B&B tree. The layer number (depth) is reported in the first column of these tables. At each layer, the LR classifier is trained based on the projection cone of the CGLP that is solved at the nodes of that layer. The training time is reported in the second column of Tables 9 and 10. Then, for each unpruned node in that layer, the classification is performed on the vector defined by the fixed partial assignment in that node used in the objective function of the CGLP; see Section 4.2 for a detailed account on the derivation procedure and algorithmic settings. The classification results are compared with the true values obtained from solving the CGLP at each node. Columns 3-6 in Tables 9 and 10 represent the elements of the confusion matrix for all the nodes in the layer. The accuracy of the classification approach for each layer is reported in the last column.

In view of the results of Tables 9 and 10, we observe that the overall accuracy often decreases as the B&B tree depth increases. Note that size of the the CGLP formed for each node of the B&B tree decreases as the depth of that node increases due to the fact that the number of unfixed variables used in Algorithm 1

to multiply with the constraints of the original problem decreases. This leads to a CGLP model with fewer variables and constraints. Therefore, as observed in the CHM results, the classification approach yields a lower accuracy for smaller problem sizes. Nevertheless, the reduction in accuracy is mostly attributed to the increase in the false negative misclassification, as a result of which a cut is not added to the model. This misclassification does not lead to excluding feasible solutions. Furthermore, the above-mentioned decrease in the size of the underlying CGLPs also leads to a smaller training time as the depth of the B&B tree increases. Lastly, we note that the total number of nodes reported in the confusion matrix entries in Tables 9 and 10 is smaller than the size of the B&B tree reported in Tables 2 and 3 since the LR method is only applied to the nodes that are not pruned by the solver after creation, which are still counted to calculate the total B&B tree size.

Table 4 Classification results for the *Random Forest* with tree number $k = 18$

Size	#	Data	TN	FP	FN	TP	Accuracy	ML Time	CGLP Time
100×300	1	train	800	0	3	797	0.99	0.36	-
		test	200	50	51	199	0.79	1.99e-5	0.01
	2	train	800	0	8	792	0.99	0.32	-
		test	185	65	59	191	0.75	2.19e-5	0.01
	3	train	800	0	5	795	0.99	0.33	-
		test	231	19	45	205	0.87	1.39e-5	0.01
	4	train	799	1	1	799	0.99	0.34	-
		test	209	41	41	209	0.83	1.39e-5	0.01
	5	train	800	0	1	799	0.99	0.32	-
		test	223	27	56	194	0.83	1.39e-5	0.01
200×1000	1	train	2400	0	9	2391	0.99	1.99	-
		test	607	93	146	554	0.82	1.42e-5	0.09
	2	train	2400	0	11	2389	0.99	2.02	-
		test	588	112	128	572	0.82	1.14e-5	0.08
	3	train	2400	0	6	2394	0.99	1.59	-
		test	690	10	128	572	0.90	1.14e-6	0.08
	4	train	2400	0	8	2392	0.99	1.81	-
		test	586	114	166	534	0.8	7.14e-6	0.09
	5	train	2400	0	4	2396	0.99	2.04	-
		test	688	12	149	2396	0.88	1.21e-5	0.09
500×2000	1	train	5000	0	7	4993	0.99	6.42	-
		test	824	676	266	1234	0.68	1.70e-5	1.28
	2	train	5000	0	12	4988	0.99	7.19	-
		test	1298	202	271	1229	0.84	1.59e-5	0.36
	3	train	5000	0	4	4996	0.99	3.48	-
		test	1310	190	269	1231	0.84	8.00e-6	0.36
	4	train	5000	0	10	4990	0.99	2.93	-
		test	1372	128	284	1216	0.86	7.33e-6	0.36
	5	train	5000	0	8	4992	0.99	2.91	-
		test	1080	420	245	1255	0.77	1.06e-5	0.36

Table 5 **Classification results for the K-*Nearest Neighbor* with number of neighbors $k = 4$**

Size	#	Data	TN	FP	FN	TP	Accuracy	ML Time	CGLP Time
100×300	1	train	800	0	435	365	0.72	0.03	-
		test	250	0	186	64	0.62	0.002	0.01
	2	train	800	0	426	374	0.73	0.03	-
		test	243	7	196	54	0.59	0.002	0.01
	3	train	800	0	428	372	0.73	0.03	-
		test	247	3	207	43	0.58	0.002	0.01
	4	train	800	0	445	355	0.72	0.03	-
		test	249	1	197	53	0.60	0.002	0.01
	5	train	800	0	419	381	0.73	0.03	-
		test	250	0	193	57	0.61	0.002	0.01
200×1000	1	train	2400	0	966	1434	0.79	0.45	-
		test	700	0	553	147	0.60	0.03	0.09
	2	train	2400	0	1002	1398	0.79	0.46	-
		test	699	1	531	169	0.62	0.03	0.08
	3	train	2400	0	994	1406	0.79	0.46	-
		test	676	24	539	161	0.59	0.02	0.08
	4	train	2400	0	1014	1386	0.78	0.41	-
		test	667	33	555	145	0.58	0.02	0.09
	5	train	2400	0	1004	1396	0.81	0.42	-
		test	699	1	543	157	0.61	0.02	0.09
500×2000	1	train	5000	0	796	4204	0.92	2.84	-
		test	1471	29	1191	309	0.59	0.11	1.28
	2	train	5000	0	802	4198	0.91	2.96	-
		test	1448	52	1183	317	0.58	0.06	0.36
	3	train	5000	0	826	4174	0.91	1.41	-
		test	1496	4	1198	302	0.59	0.06	0.36
	4	train	5000	0	789	4211	0.92	1.12	-
		test	1388	12	1173	327	0.60	0.06	0.36
	5	train	5000	0	799	4201	0.92	1.11	-
		test	1407	93	1202	298	0.56	0.05	0.36

Table 6 Classification results for the *Logistic Regression* with *solver=liblinear*, inverse of regularization strength $C = 1$, and probability $P = 0.5$

Size	#	Data	TN	FP	FN	TP	Accuracy	ML Time	CGLP Time
100×300	1	train	651	149	1	799	0.90	0.09	-
		test	250	0	1	249	0.99	0.0	0.01
	2	train	642	158	0	800	0.90	0.09	-
		test	250	0	1	249	0.99	1.99e-6	0.01
	3	train	635	165	0	800	0.89	0.09	-
		test	250	0	0	250	1	2.00e-6	0.01
	4	train	644	156	1	799	0.90	0.09	-
		test	250	0	0	250	1	4.00e-6	0.01
	5	train	648	152	1	799	0.90	0.10	-
		test	350	0	1	249	0.99	2.00e-6	0.01
200×1000	1	train	2212	188	0	2400	0.96	1.16	-
		test	700	0	0	700	1	4.99e-6	0.09
	2	train	2189	211	0	2400	0.95	1.22	-
		test	700	0	2	698	0.99	2.85e-6	0.08
	3	train	2186	214	0	2400	0.95	1.07	-
		test	700	0	1	699	0.99	6.42e-6	0.08
	4	train	2173	227	0	2400	0.95	0.96	-
		test	700	0	0	700	1	3.57e-6	0.09
	5	train	2195	205	0	2400	0.95	0.96	-
		test	700	0	0	700	1	3.57e-6	0.09
500×2000	1	train	4942	58	0	5000	0.99	5.30	-
		test	1500	0	4	1496	0.99	6.33e-6	1.28
	2	train	4960	40	0	5000	0.99	3.62	-
		test	1500	0	3	1497	0.99	7.66e-6	0.36
	3	train	4948	52	0	5000	0.99	3.12	-
		test	1500	0	6	1494	0.99	4.99e-6	0.36
	4	train	4975	25	0	5000	0.99	3.31	-
		test	1500	0	9	1491	0.99	4.33e-6	0.36
	5	train	4924	76	0	5000	0.99	3.03	-
		test	1500	0	9	1491	0.99	3.99e-6	0.36

Table 7 **Classification results for the *Support Vector Machine* with *kernel=linear* and regularization parameter $C = 0.4$**

Size	#	Data	TN	FP	FN	TP	Accuracy	ML Time	CGLP Time
100×300	1	train	602	198	0	800	0.87	0.70	-
		test	250	0	0	250	1	0.0003	0.01
	2	train	600	200	0	800	0.87	0.71	-
		test	250	0	0	250	1	0.0003	0.01
	3	train	602	198	0	800	0.87	0.67	-
		test	250	0	0	250	1	0.0003	0.01
	4	train	600	200	0	800	0.87	0.66	-
		test	250	0	0	250	1	0.0003	0.01
	5	train	606	194	0	800	0.87	0.66	-
		test	250	0	0	250	1	0.0003	0.01
200×1000	1	train	2346	54	0	2400	0.98	15.53	-
		test	700	0	5	695	0.99	0.003	0.09
	2	train	2342	58	0	2400	0.98	13.91	-
		test	700	0	9	691	0.99	0.002	0.08
	3	train	2317	83	0	2400	0.98	13.96	-
		test	700	0	13	687	0.99	0.002	0.08
	4	train	2304	96	0	2400	0.98	13.24	-
		test	700	0	4	696	0.99	0.002	0.09
	5	train	2317	83	0	2400	0.98	12.90	-
		test	700	0	6	694	0.99	0.002	0.09
500×2000	1	train	5000	0	2	4998	0.99	100.63	-
		test	1500	0	84	1416	0.97	0.009	1.28
	2	train	5000	0	5	4995	0.99	73.96	-
		test	1500	0	109	1391	0.96	0.005	0.36
	3	train	5000	0	2	4998	0.99	51.58	-
		test	1500	0	95	1405	0.96	0.004	0.36
	4	train	5000	0	6	4994	0.99	59.99	-
		test	1500	0	118	1382	0.96	0.004	0.36
	5	train	4998	2	3	4997	0.99	52.90	-
		test	1500	0	75	1425	0.97	0.005	0.36

Table 8 **Classification results for the *Neural Network* with *hidden layer* $= 18$, *activation function*= *identity*, *solver* = *sgd*, *batch size* $= 100$, *regularization parameter* $\alpha = 0.01$, *learning rate* = *adaptive*, *initial learning rate* $= 0.0007$, and *maximum number of iterations* $= 1500$**

Size	#	Data	TN	FP	FN	TP	Accuracy	ML Time	CGLP Time
100×300	1	train	696	104	6	794	0.93	12.88	-
		test	250	0	6	244	0.98	4.00e-6	0.01
	2	train	712	88	2	798	0.94	14.36	-
		test	250	0	5	245	0.99	4.00e-6	0.01
	3	train	712	88	3	797	0.94	14.37	-
		test	250	0	6	244	0.98	1.99e-6	0.01
	4	train	704	96	9	791	0.93	13.25	-
		test	250	0	8	242	0.98	2.00e-6	0.01
	5	train	708	92	8	792	0.93	13.71	-
		test	250	0	8	242	0.98	1.99e-6	0.01
200×1000	1	train	2380	20	1	2399	0.99	65.60	-
		test	700	0	7	693	0.99	6.42e-6	0.09
	2	train	2383	17	2	2398	0.99	69.05	-
		test	700	0	16	684	0.98	4.28e-6	0.08
	3	train	2376	24	1	2399	0.99	62.59	-
		test	700	0	19	681	0.98	3.57e-6	0.08
	4	train	2367	33	1	2399	0.99	65.45	-
		test	700	0	14	686	0.99	6.42e-6	0.09
	5	train	2367	33	4	2396	0.99	63.31	-
		test	700	0	9	691	0.99	5.00e-6	0.09
500×2000	1	train	4994	6	1	4999	0.99	158.87	-
		test	1500	0	25	1475	0.99	7.66e-6	1.28
	2	train	4998	2	2	4998	0.99	111.11	-
		test	1500	0	31	1469	0.98	8.33e-6	0.36
	3	train	4995	5	2	4998	0.99	110.06	-
		test	1500	0	24	1476	0.99	5.33e-6	0.36
	4	train	4999	1	3	4997	0.99	86.85	-
		test	1500	0	31	1469	0.98	4.66e-6	0.36
	5	train	4987	13	2	4998	0.99	92.90	-
		test	1500	0	36	1464	0.98	5.33e-6	0.36

Table 9 Classification results for each layer of the B&B tree for the first instance of Table 2

B&B Tree Depth	Train Time	True Negative	False Positive	False Negative	True Positive	Accuracy
1	20.83	2	0	0	0	1.00
2	20.42	2	0	0	2	1.00
3	19.42	2	0	0	2	1.00
4	18.19	2	0	0	2	1.00
5	17.75	4	0	0	0	1.00
6	16.72	7	0	0	1	1.00
7	15.54	9	1	0	4	0.93
8	15.10	10	1	1	6	0.89
9	13.53	11	0	1	10	0.95
10	12.76	20	0	2	2	0.92
11	12.53	20	0	3	16	0.92
12	11.63	20	2	3	17	0.88
13	10.70	22	0	4	17	0.91
14	10.44	22	0	4	19	0.91
15	9.64	23	2	5	18	0.85
16	8.67	30	0	9	13	0.83
17	8.09	26	1	14	35	0.80
18	7.34	34	0	23	22	0.71
19	6.65	33	0	24	21	0.69
20	5.99	23	1	47	30	0.52
21	5.45	38	0	31	12	0.62
22	4.58	52	0	33	15	0.67
23	3.40	44	0	54	36	0.60
24	2.97	38	0	62	10	0.44
25	2.58	46	0	51	17	0.55
26	2.27	44	1	46	18	0.57
27	1.93	53	0	42	0	0.56
28	1.65	43	0	54	0	0.44
29	1.40	23	0	56	0	0.29
30	1.15	41	0	35	0	0.54
31	0.95	36	0	29	0	0.55
32	0.80	33	0	22	0	0.60
33	0.61	17	0	15	0	0.53
34	0.40	17	0	9	0	0.65
35	0.33	7	0	8	0	0.47
36	0.18	4	0	2	0	0.67
37	0.10	4	0	1	0	0.80
38	0.04	1	0	0	0	1.00

Table 10 Classification results for each layer of the B&B tree for the first instance of Table 3

B&B Tree Depth	Train Time	True Negative	False Positive	False Negative	True Positive	Accuracy
1	6.0372	2	0	0	0	1.00
2	5.5961	3	0	0	1	1.00
3	5.1293	3	0	1	2	0.83
4	4.9456	4	1	0	3	0.88
5	4.7717	5	0	1	2	0.88
6	4.5564	7	0	1	1	0.89
7	4.0524	6	1	2	4	0.77
8	3.8808	7	0	4	3	0.71
9	3.7526	10	0	3	7	0.85
10	3.3934	12	2	5	6	0.72
11	2.9123	14	0	8	8	0.73
12	2.6759	17	1	12	6	0.64
13	2.4469	12	2	13	10	0.59
14	2.2926	19	0	13	4	0.64
15	2.0172	23	0	22	13	0.62
16	1.8169	31	0	23	21	0.69
17	1.6957	29	1	36	24	0.59
18	1.6244	32	0	41	20	0.56
19	1.4567	35	0	44	12	0.52
20	1.2023	31	0	53	20	0.48
21	0.7969	26	0	40	14	0.50
22	0.6471	24	0	38	10	0.47
23	0.5648	28	0	31	7	0.53
24	0.4168	26	0	26	8	0.57
25	0.3536	23	0	22	4	0.55
26	0.2855	20	0	15	0	0.57
27	0.1453	17	0	8	0	0.68
28	0.1172	14	0	5	0	0.74
29	0.0871	7	0	2	0	0.78
30	0.0694	5	0	1	0	0.83

Appendix B: Sensitivity Analysis

As noted in Section (4.1), one set of class-1 vectors in the training data set is generated based on the result of Corollary 5(i), which requires non-redundancy of constraints of $\mathcal{C}$. Since checking redundancy of constraints in practical applications is time-consuming, we perform sensitivity analysis on the redundancy ratio of constraints to evaluate its impact on classification results. For this analysis, we study different redundancy ratios as in Figure 22 of Kalantari and Zhang (2022) by creating redundant vectors through a conic combination of vectors of $\mathcal{C}$ and adding them to the cone. We conduct these experiments on the first instance of each size category. These results for the top three classifiers LR, SVM and NN are summarized in Figure 3. As shown in these figures, the classification accuracy for training data declines slightly as the redundancy ratio increases. This decline is due to the fact that the training data generated through Corollary 5(i) are not guaranteed to be a class-1 vector when the base constraint is redundant. These vectors, however, are labeled as class-1 in the training set, leading to a potential misclassification. Nevertheless, it is inferred from these experiments that the impact of this source of misclassification is reduced due to the balance in the remaining sets of training data, which results in maintaining high accuracy of the test set across different redundancy ratios and problem sizes.

Another deciding factor in creating training data set is ϵ, which determines the proximity of class-0 and class-1 vectors to the boundary of the polar cone. To investigate the marginal impact of different values of ϵ on the classification performance, we next conduct sensitivity analysis on the first instance of each problem size category and each classification method. These results for the top three classifiers LR, SVM and NN are summarized in Figure 4. As observed in these figures, the common trend across all classification methods and problem sizes is that, increasing the ϵ value leads to a higher training accuracy and a lower test accuracy. This trend is expected as the larger ϵ values broaden the spatial gap between class-0 and class-1 training data, which enables the classifier to improve training accuracy at the expense of reducing test accuracy.

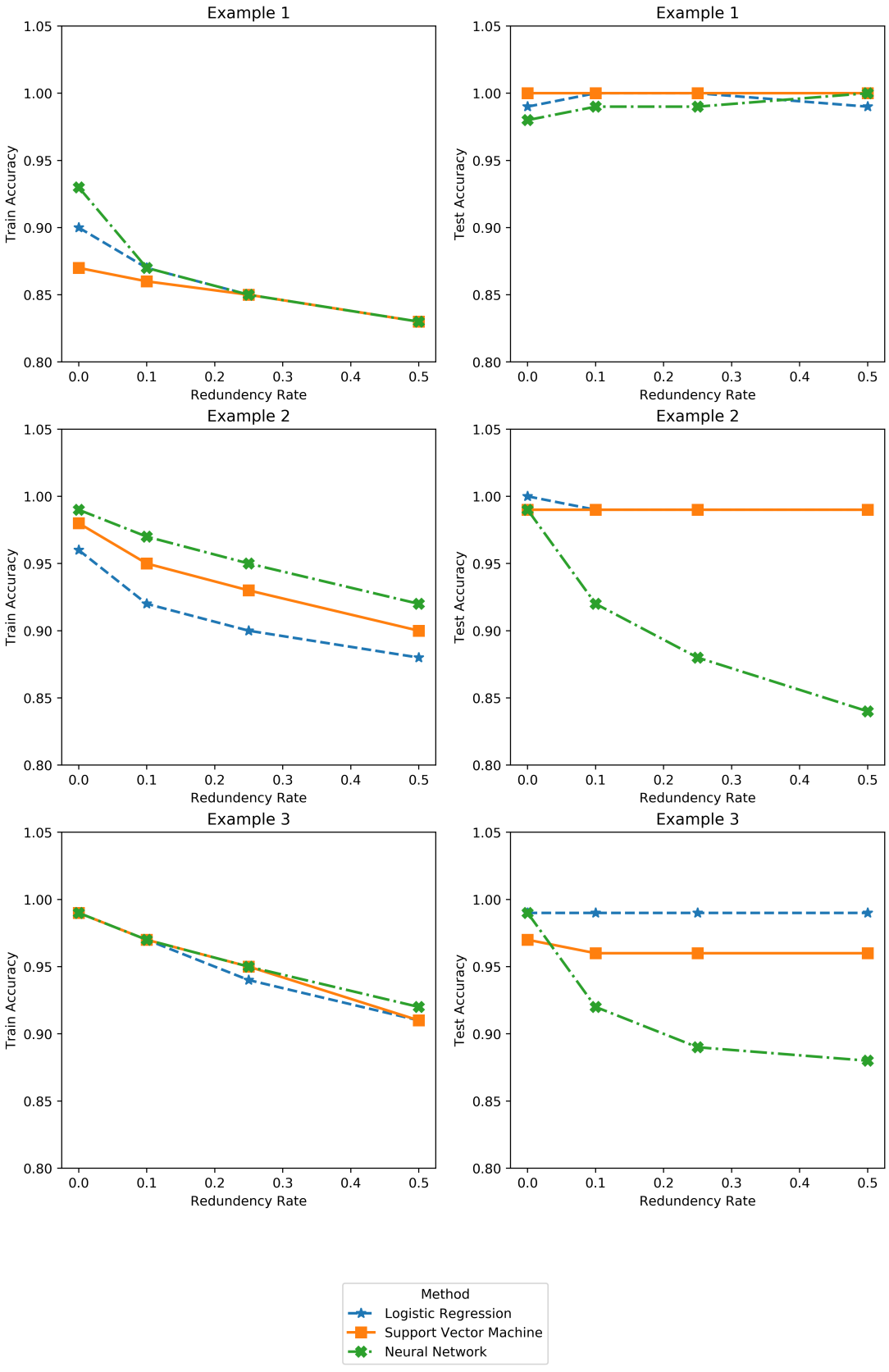

Figure 3 **Sensitivity analysis results for redundancy ratio**

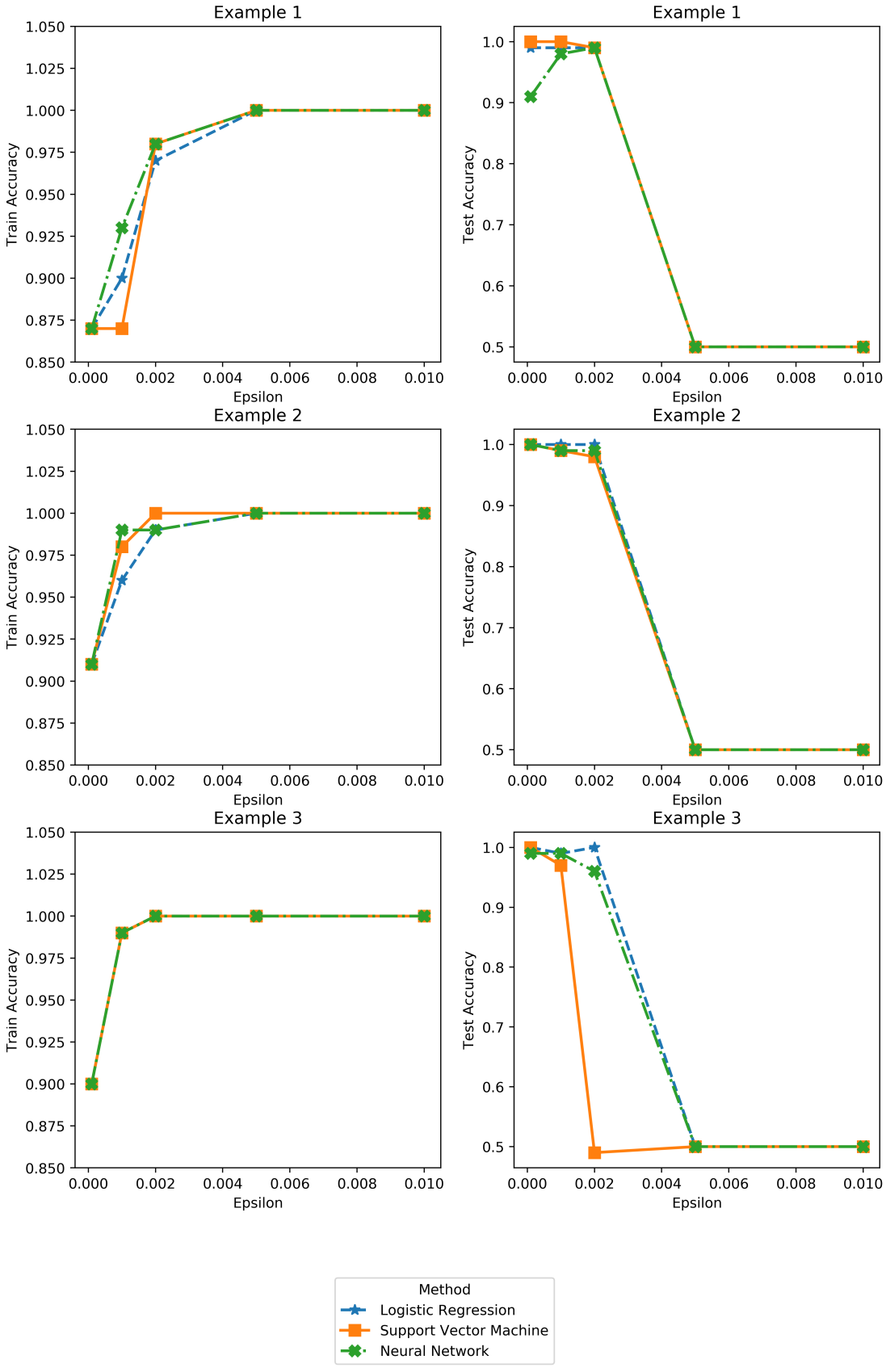

Figure 4 **Sensitivity analysis results for values of ϵ**

As noted in Section 4.1, we have used $4(m+n)$ data points in our training set for the CHM problem. In Figures 5 and 6, we show the relationship between the size of the training set and the training time and accuracy of the classification for the LR method. We conduct these experiments on one instance of each size category (100×300), (200×1000), and (500×2000). We consider four different sizes for the training set as multiples of the total number of constraints in the cone $\mathcal{C}$, namely $2(m+n)$, $4(m+n)$, $6(m+n)$, and $8(m+n)$. All the other parameters are set similarly to those used in the experiments reported in Section 4.1. As observed in these figures, increasing the size of the training set leads to increasing the accuracy at the price of increasing the training time. Based on these results, the choice of $4(m+n)$ provides a reasonable trade-off between the training time and accuracy.

Figure 5 **The impact of the size of the training set on the training time for the LR method in CHM problem.**

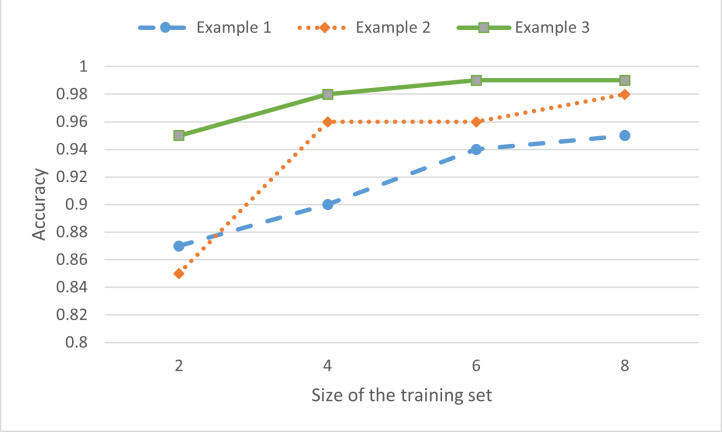

Figure 6 **The impact of the size of the training set on the training accuracy for the LR method in CHM problem.**

For the next sensitivity analysis, we consider the test accuracy for the LR method applied to the CHM problem for different categories of test instances based on their distance from the boundary of the polar cone. To this end, we use similar settings to those mentioned in Section 4.1 to train a LR classifier for one

Table 11 Classification results for the *Logistic Regression* approach for test instances with different distance values from the boundary of the polar cone

Size	ϵ	TN	FP	FN	TP	Accuracy
100×300	0.0005	774	26	190	610	0.87
	0.001	793	7	149	651	0.91
	0.0015	800	0	94	706	0.95
	0.002	800	0	61	739	0.98
	0.0025	800	0	15	785	0.99
200×1000	0.0005	2359	41	394	2006	0.91
	0.001	2385	15	191	2209	0.94
	0.0015	2400	0	117	2283	0.98
	0.002	2400	0	46	2354	0.99
	0.0025	2400	0	0	2400	1
500×2000	0.0005	5000	0	855	4145	0.92
	0.001	5000	0	504	4496	0.96
	0.0015	5000	0	57	4943	0.99
	0.002	5000	0	0	5000	1
	0.0025	5000	0	0	5000	1

instance of each size category. Then, we use the resulting classifier to predict the class of data points in the test sets that are generated based on a similar method used to generate the training set where the proximity of the generated vector is determined by the ϵ value in Corollary 1 for class-0 and Corollary 5(i) for class-1. As noted in Section 4.1, the vectors produced in this process are normalized to have a uniform scaling for all data points in class-1 and class-0. We consider five categories for $\epsilon \in \{0.0005, 0.001, 0.0015, 0.002, 0.0025\}$ to represent different distance values. For each distance category of the problem size (100×300), (200×1000), and (500×2000), we consider the test set size 1600, 4800, and 10000, respectively. The accuracy results, including the elements of the confusion matrix, are given in Table 11 and Figure 7. As observed from these plots, the test accuracy increases with the distance of the data points from the boundary of the polar cone.

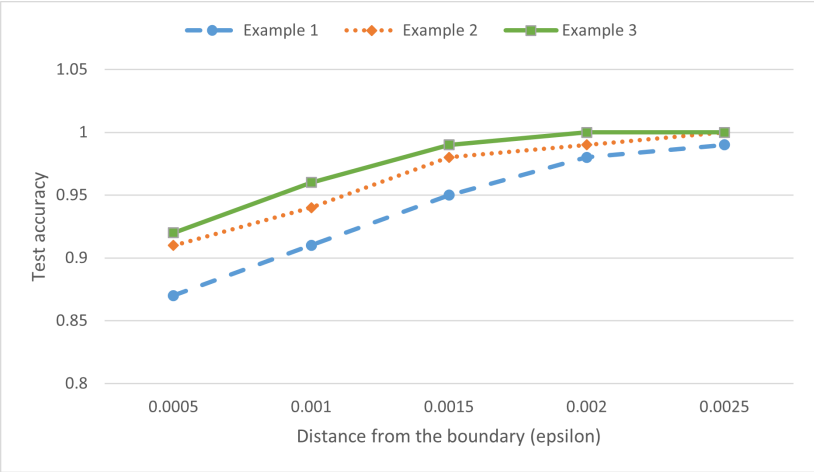

Figure 7 **The test accuracy for different distance categories for the LR method in CHM problem.**

Appendix C: Consistency Cuts and Strong Branching

As mentioned in Section 3, the goal of using consistency cuts is to achieve (partial) LP-consistency. An alternative approach to ensure partial-LP consistency for rank r at any node of the B&B tree associated with the partial assignment $\boldsymbol{\alpha}_I = \boldsymbol{v}_I$ such that $\mathcal{S}_{\text{LP}} \cap \{\boldsymbol{\alpha} \in \mathbb{R}^n : \boldsymbol{\alpha}_I = \boldsymbol{v}_I\} \neq \emptyset$ is to check whether there exists $J \subseteq N \setminus I$ with $|J| = r$ such that the LP relaxations described by $\mathcal{S}_{\text{LP}} \cap \{\boldsymbol{\alpha} \in \mathbb{R}^n : \boldsymbol{\alpha}_{I\cup J} = \boldsymbol{v}_{I\cup J}\}$ is infeasible for every 0–1 value assignment $\boldsymbol{\alpha}_J = \boldsymbol{v}_J$. In this case, the node will be pruned. This approach is similar to the so-called *strong branching* technique frequently used in B&B by solvers. It is shown in Davarnia et al. (2022) that the consistency cut framework has a computational advantage compared to the strong branching method as outlined next. In particular, Proposition 5.5 in Davarnia et al. (2022) shows that applying the CGLP of Proposition 6 for a certain rank r can achieve LP-consistency of ranks higher than r. In other words, the CGLP obtained from the intersection of the multilinear constraints produced by the multiplication with unfixed variables can lead to separating inconsistent faces that remain feasible if we consider the disjunctive model for each unfixed variable individually (which is equivalent to the outcome of the strong branching); see Example 5.3 in Davarnia et al. (2022) for an illustration. This property can lead to a significant reduction in the B&B tree size when using the CGLP approach compared to the strong branching approach. As a numerical evidence for the above property, we present in Tables 12 and 13 computational experiments on the instances of Tables 2 and 3, respectively, which show that the outcome of the consistency approach outperforms that of the strong branching method in terms of both the tree size and the solution time.

Appendix D: Machine Learning Approach for Higher Consistency Ranks

In this section, we present computational results to evaluate the performance of the ML approach in approximating the CGLP corresponding to partial LP-consistency of higher ranks compared to that of rank one presented in Table 2 of Section 4.2. To obtain the CGLP for rank r, following Algorithm 1, we need to multiply each constraint of the original model with all 2^r product combinations of every tuple of size r of unfixed variables at each node of the B&B tree. As a result, increasing r leads to larger problem sizes in terms of both the number of variables and constraints in the CGLP that is solved at each node of the B&B

Table 12 Comparison of the consistency cuts and the strong branching approach for multi-knapsack problems

n	m	#	B&B		CGLP				Strong Branching			
			nodes	time	nodes	Δ (%)	time	Δ (%)	nodes	Δ (%)	time	Δ (%)
40	50	1	34473	3142.60	**2383**	*93*	**1103.69**	*64*	12755	*63*	1351.06	*57*
		2	82417	7496.14	**5625**	*93*	**3548.07**	*52*	26373	*68*	4122.8	*45*
		3	78965	7167.15	**5225**	*93*	**2197.50**	*69*	25269	*68*	2436.78	*66*
		4	124327	11377.96	**5863**	*95*	**2682.24**	*76*	36055	*71*	3754.41	*67*
		5	126209	11700.49	**6197**	*95*	**3120.76**	*73*	37863	*70*	3393.05	*71*
45	55	1	1050459	119705.93	**63677**	*93*	**31929.71**	*73*	189083	*82*	49079.07	*59*
		2	57621	5449.79	**3519**	*93*	**2116.51**	*61*	17863	*69*	2288.58	*58*
		3	94281	8424.51	**4693**	*95*	**2965.73**	*64*	26399	*72*	3875.14	*54*
		4	450175	44544.77	**22645**	*94*	**13808.84**	*69*	112543	*75*	18708.34	*58*
		5	630449	65030.67	**44875**	*92*	**26270.06**	*59*	176525	*71*	27962.9	*57*
45	60	1	989029	116212.23	**54721**	*94*	**35255.78**	*69*	326379	*67*	48809.14	*58*
		2	257217	25121.01	**17321**	*93*	**9948.30**	*60*	56587	*78*	11304.45	*55*
		3	249869	23936.52	**14063**	*94*	**7917.20**	*66*	59969	*76*	9335.40	*61*
		4	-	>86400	**56847**	-	**31665.33**	*>63*	239325	-	38016.84	*>56*
		5	320967	31300.88	**17925**	*94*	**12102.74**	*61*	80241	*75*	13772.13	*56*
50	60	1	329277	31947.91	**12393**	*96*	**11852.70**	*62*	8945	*72*	14696.62	*54*
		2	531833	54939.26	**22885**	*95*	**20104.97**	*63*	106367	*80*	21975.36	*60*
		3	-	>86400	**93375**	-	**70096.13**	*>18*	103369	-	76896.19	*>11*
		4	174357	16540.17	**8533**	*95*	**7478.74**	*54*	41845	*76*	9262.45	*44*
		5	-	>86400	**53671**	-	**44288.28**	*>48*	43507	-	50112.72	*>42*

Table 13 Comparison of the consistency cuts and the strong branching approach for MIPLIB problems

Class	n	m	B&B		CGLP				Strong Branching			
			nodes	time	nodes	Δ (%)	time	Δ (%)	nodes	Δ (%)	time	Δ (%)
`p0033`	33	15	32117	2961.55	**385**	*98*	**283.14**	*90*	5139	*84*	887.3	*70*
`pipex`	41	48	1057	1933.48	**623**	*41*	**1484.10**	*23*	707	*33*	1527.07	*19*
`sentoy`	60	30	703	391.12	**271**	*63*	**187.27**	*53*	365	*48*	218.96	*44*
`stein27`	27	118	6099	2474.1	**4823**	*22*	2115.07	*15*	4819	*21*	**1927.38**	*22*
`enigma`	100	42	98545	7413.51	**21679**	*78*	**2669.5**	*64*	55185	*44*	3632.37	*51*
`lseu`	89	28	219597	4138.8	**186963**	*15*	**3724.2**	*10*	193245	*12*	3806.96	*8*

tree, which leads to an extensive computational burden. For example, for most of the instances studied in Table 2, we ran into memory errors when using the CGLP of rank $r = 2$. As an example, when there are 10 unfixed variables remaining at a layer of the B&B tree, to obtain the CGLP for rank $r = 1$, the number of

constraints in the linearized system will be multiplied by 20, whereas for CGLP of rank $r = 2$, this size will be multiplied by 180. Nonetheless, to provide some insight on the performance of the ML approach for ranks higher than one, we present in Table 14 computational experiments that compare the outcome of using ML for rank $r = 1$ and $r = 2$ for instances of Section 4.2.

For these experiments, due to extensive computational burden of applying the consistency framework for higher ranks at each node of the B&B tree, as discussed above, we have used the common (K, L) approach described as follows. The goal is to reduce the implementation time by (i) applying the CGLP (or its alternative ML approach in this case) in certain layers of the B&B tree only, and (ii) choosing a subset of unfixed variables to multiply with the constraints. We represent the layer candidates for applying the CGLP by K, and the variable indices used for multiplication by L. It is intuitive to pick top layers of the B&B tree to be included in K, and choose variable indices at the bottom layers of the tree to be included in L; see Davarnia et al. (2022) for further illustration of this approach. For the experiments presented in Tables 14 and 15, we have chosen $K = 10$ and $L = 10$ for both ML approaches targeting partial LP-consistency of rank one and two to provide a fair comparison in a controlled setting. The columns in these tables are defined similarly to those of Tables 2 and 3 with a difference that the subcolumns under "LR, $r = 1$" and "LR, $r = 2$" include the B&B tree size and solution time for applying the logistic regression approach for partial LP-consistency of rank one and two using the (K, L) approach. The implementation and algorithmic settings are similar to those used in Section 4.2. All instances in these experiments have been solved to optimality for both approaches. As observed in Tables 14 and 15, the logistic regression method when used to approximate the outcome of the CGLP of rank $r = 2$ outperforms that used for rank $r = 1$ in both B&B tree size and total solution time in most of the instances.

Table 14 Comparison of the ML approach for different consistency ranks for multi-knapsack problems

n	m	#	B&B		LR, $r=1$, and $K=L=10$				LR, $r=2$, and $K=L=10$			
			nodes	time	nodes	Δ (%)	time	Δ (%)	nodes	Δ (%)	time	Δ (%)
40	50	1	34473	3142.60	26651	*23*	2453.62	*22*	**22967**	*34*	**2186.80**	*31*
		2	82417	7496.14	76177	*8*	7051.42	*6*	**54347**	*35*	**5116.49**	*32*
		3	78965	7167.15	78717	*0*	7258.27	*−1*	**69523**	*12*	**6524.29**	*9*
		4	124327	11377.96	117233	*6*	10832.11	*5*	**101065**	*19*	**9421.17**	*18*
		5	126209	11700.49	98121	*24*	9072.97	*23*	**61539**	*52*	**5772.71**	*51*
45	55	1	1050459	119705.93	1035775	*2*	116557.46	*3*	**190993**	*81*	**17972.34**	*85*
		2	57621	5449.79	53507	*8*	4265.75	*22*	**32073**	*45*	**2971.28**	*46*
		3	94281	8424.51	62925	*34*	5876.83	*30*	**12029**	*87*	**1149.06**	*86*
		4	450175	44544.77	398539	*12*	39966.08	*11*	**286923**	*37*	**27952.67**	*38*
		5	630449	65030.67	630449	*0*	65690.10	*−2*	**263239**	*58*	**25554.13**	*61*
45	60	1	989029	116212.23	951755	*4*	104927.14	*10*	**779599**	*22*	**85763.15**	*27*
		2	257217	25121.01	255045	*1*	23902.25	*5*	**207051**	*20*	**19344.72**	*24*
		3	249869	23936.52	233177	*7*	21761.88	*10*	**63619**	*74*	**5980.96**	*76*
		4	-	>86400	823355	-	90986.38	-	**641949**	-	**77075.34**	-
		5	320967	31300.88	320363	*0*	31782.85	*−2*	**182909**	*45*	**16944.11**	*46*
50	60	1	329277	31947.91	270375	*18*	25977.05	*20*	**166139**	*50*	**15501.51**	*52*
		2	531833	54939.26	516531	*3*	52111.54	*6*	**93973**	*82*	**8679.84**	*85*
		3	-	>86400	-	-	-	-	-	-	-	-
		4	174357	16540.17	171407	*2*	15523.94	*7*	**151981**	*13*	**13994.98**	*16*
		5	-	>86400	-	-	-	-	-	-	-	-

Table 15 Comparison of the ML approach for different consistency ranks for MIPLIB problems

Class	n	m	B&B		LR, $r=1$, and $K=L=10$				LR, $r=2$, and $K=L=10$			
			nodes	time	nodes	Δ (%)	time	Δ (%)	nodes	Δ (%)	time	Δ (%)
p0033	33	15	32117	2961.55	22161	*31*	2191.14	*26*	**13811**	*57*	**1658.16**	*44*
pipex	41	48	1057	1933.48	983	*7*	1875.01	*3*	**865**	*18*	**1643.5**	*15*
sentoy	60	30	703	391.12	611	*13*	336.16	*14*	**505**	*28*	**273.7**	*30*
stein27	27	118	6099	2474.1	5001	*18*	**2152.37**	*13*	**4879**	*20*	2251.34	*9*
enigma	100	42	98545	7413.51	70951	*28*	5782.14	*22*	**57155**	*42*	**4077.15**	*45*
lseu	89	28	219597	4138.8	217401	*1*	4183.11	*−1*	**199833**	*9*	**3641.44**	*12*