Experiments

Chunheng Jiang

July 2016

1 Introduction

Here are some results from using neural network to learn a voting rule with specific axiomatic properties. The settings are listed in Table 1.

Table 1: Experimental Settings

Notations	Value	Description
m	3	Number of candidates
n	{5, 7, 9, 11, 13}	Number of voters
d	{9, 9, 6}	Dimension of features: positional, pairwise, m!-preference
$\mathcal{A}$	{RF, NN}	Algorithms to learn a voting rule
W	$(\omega, 1-\omega), 0 \leq \omega \leq 1$	Importance vector of RF and NN in ensemble model
nTree	5	Number of trees in RF
depth	5	Depth of decision tree in RF
nFeature	$\lfloor\sqrt{d}\rfloor$	Number of features randomly selected by tree for splitting
E	GINI	Criterion for splitting
nLayer	2	Number of layers(input layer exclusive)
nInput	d	Number of input nodes
nHidden	20	Number of hidden nodes
nOutput	m	Number of output nodes
σ	{ReLu, SoftMax}	Activation functions
L	Neg-Loglikelihood	Loss (or cost) function
OA	SGD	Optimization algorithm
α	0.001	Learning rate for network weights-updating
η	0.98	Momentum for network weights-updating
λ	0.005 * α	ℓ_1–regularization coefficient

To learn a voting rule, we extract all positional features of each candidates, e.g., the time each candidate is ranked first by voters, the votes each of them received at the second place, etc, all pairwise comparison results between any two candidates, and the distribution of the $m!$ possible preference rankings in a preference profile.

Traditional random forest techniques grow trees based on the splitting of single feature and are very suitable to learn pairwise comparison rule. However, it's not easy to approximate scoring voting rules, which is a linear combination of the single features. To improve the performance of decision tree on scoring voting rules, we could extend the splitting range from univariate to multivariate. The later requires to search from a larger space the local optimal weight vector (features indices and their weights).

Artificial neural network is very excellent in extracting potential features combination. Meanwhile, it's well-known as a black-box, with many parameters and hyper-parameters to tune. To optimize a neural network is said being more an art than a science.

We conduct a series of experiments over the training set sampling from Borda rule labeled data, profiles with Condorcet winners, and also constructed based on neutrality criterion and monotonicity criterion. We expect the learned rule could be as simple as possible, be closely approximated to the ground-truth voting rule (here is Borda rule), and could keep a high satisfiability to the specific axiomatic properties at the same time.

Random forest - easy to explain without much tuning, and neutral network - helpful to extract some potential features are combined to learn an ensemble voting function to make prediction. As shown in Fig. 1-16, each figure contains five subplots.

- The leftmost subplot shows the training accuracy of the learned rule over the training samples, including four subsamples: Borda rule labeled profiles, Condorcet criterion-based (CWC), neutrality criterion-based (Neutral) and monotonicity criterion-based (Mono) profile, and all of them as a whole.
- The second subplot indicates the approximation of the learned rule to the ground-truth voting rule, and the higher the scores the better the fitness. The label for each bar, for example the second one $n = 9$ means that the rule is evaluated on all possible preference profiles by 9 voters. Suppose the learned rule could always get a high score on each voting context with more than 3 voters, it implies that the ground truth rule has a higher probability of being learned by the algorithm. The digits are the related numbers of samples.
- Each of the right three subplots represents a corresponding axiomatic properties, they are CWC, Neutral and Mono. They tell us the differences between the satisfiabilities of the learned rule and the ground-truth rule to the related fairness criteria.

1.1 Performance of Random Forest - see Fig. 1-4

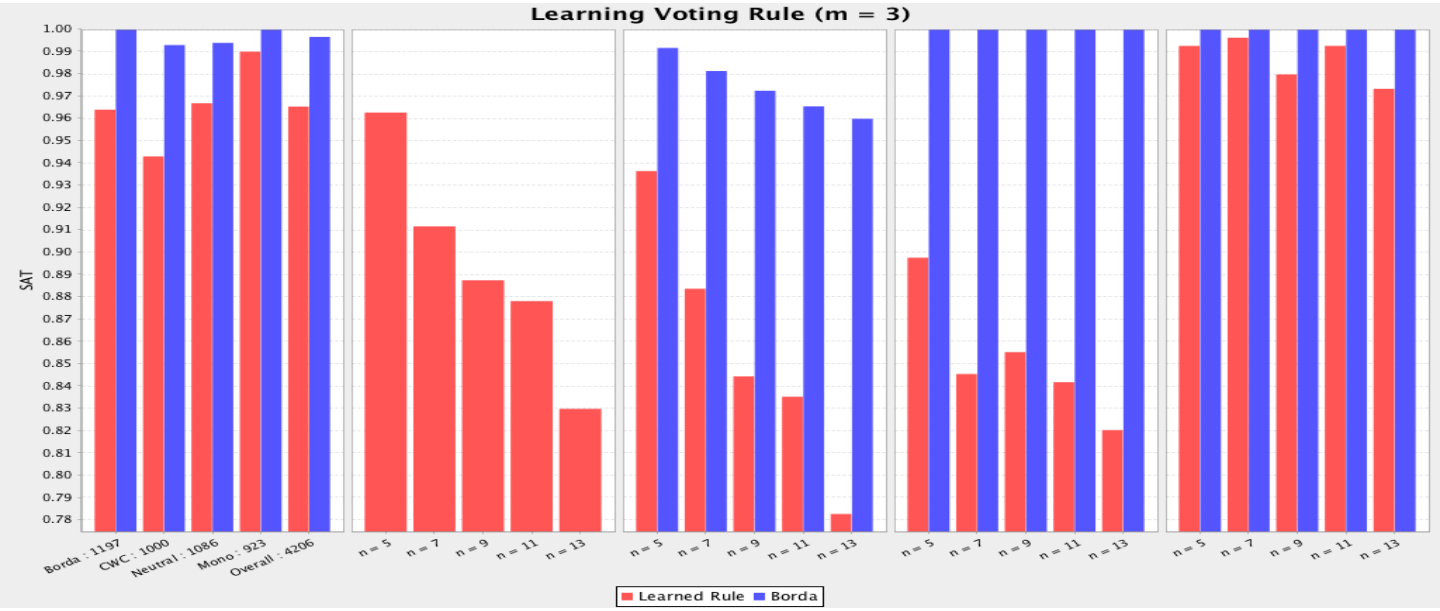

Figure 1: Performance of RF on the positional features

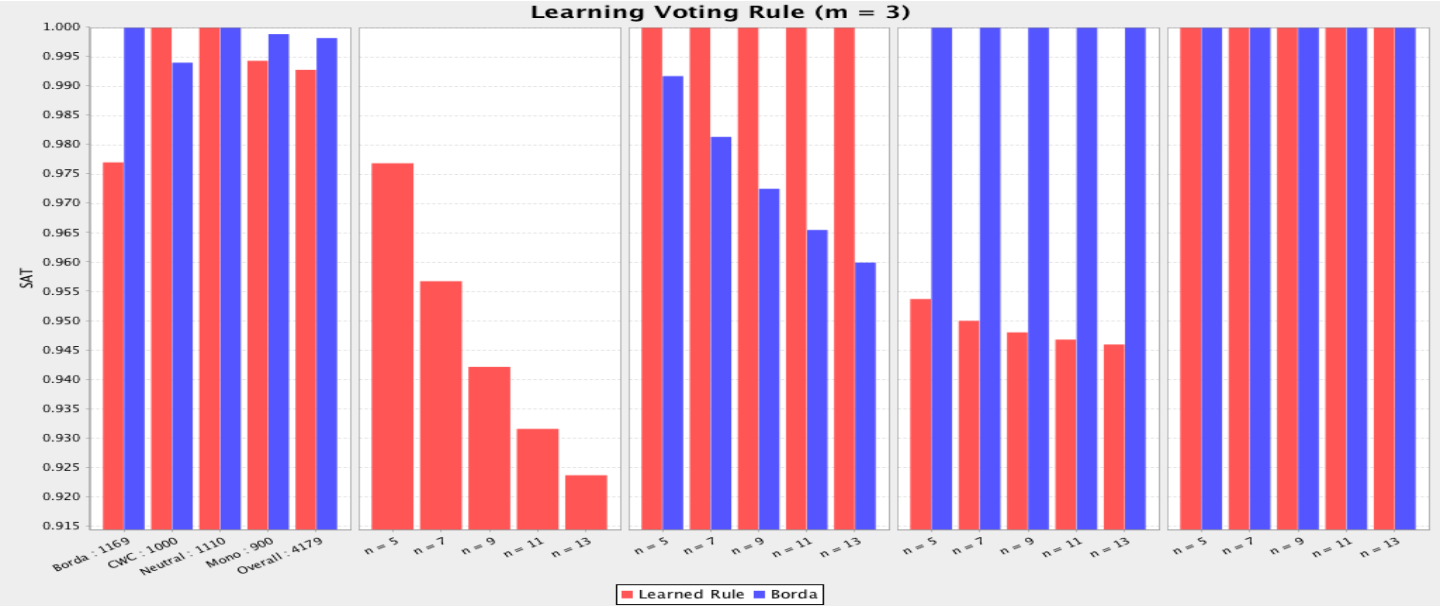

Figure 2: Performance of RF on the pairwise features

1.2 Performance of Neural Network - see Fig. 5-8

1.3 Performance of Ensemble Model: RFDL - see Fig. 9-14

1.4 Performance of Ensemble Model: RFDL with Smaller Samples - see Fig. 15-16

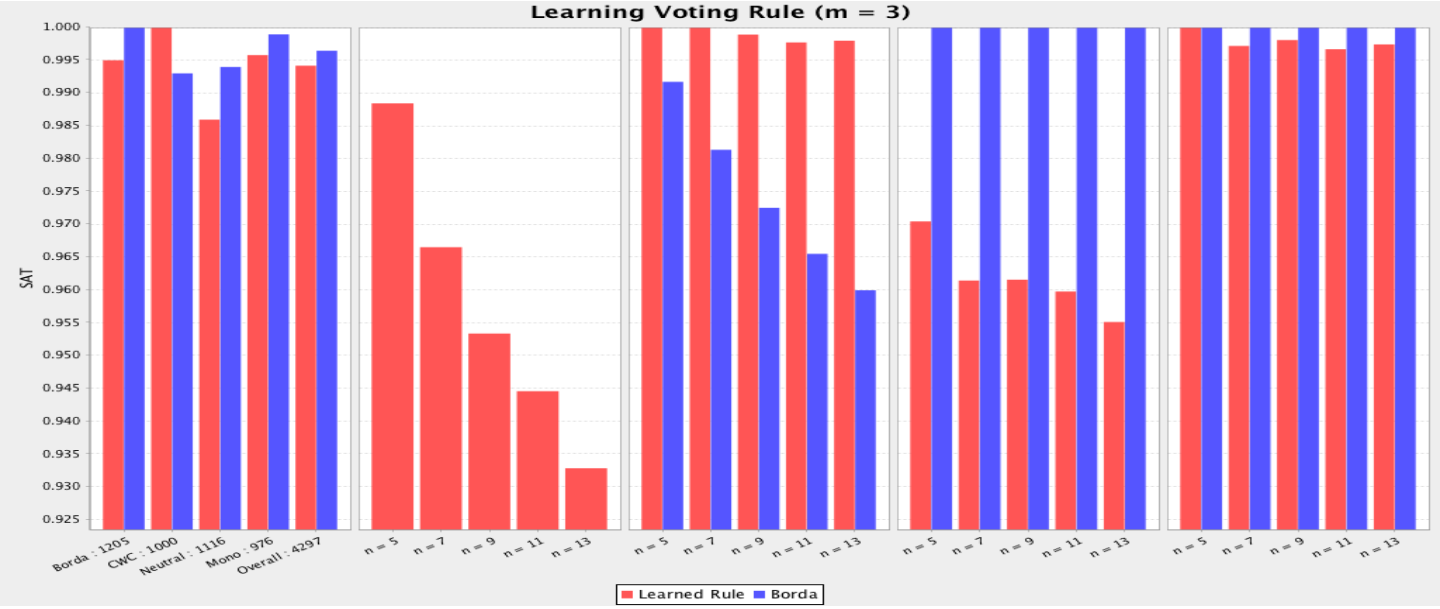

Figure 3: Performance of RF on both positional and pairwise features

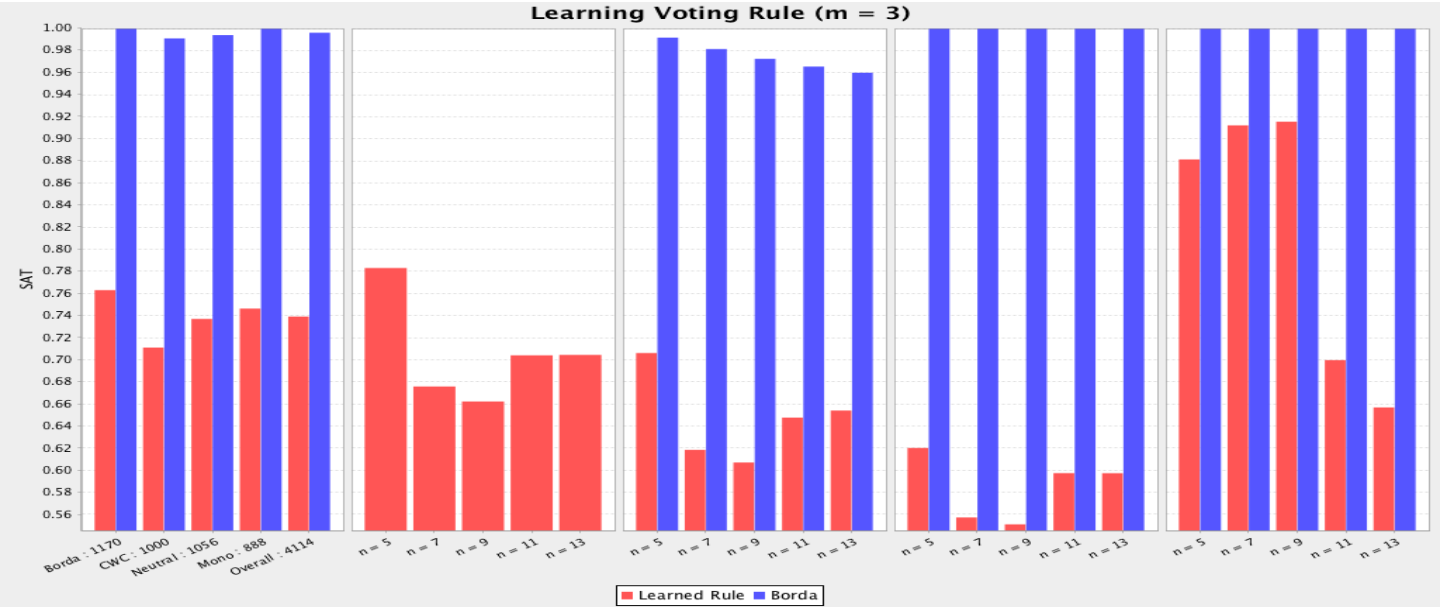

Figure 4: Performance of RF on the $m!$-preference profiles

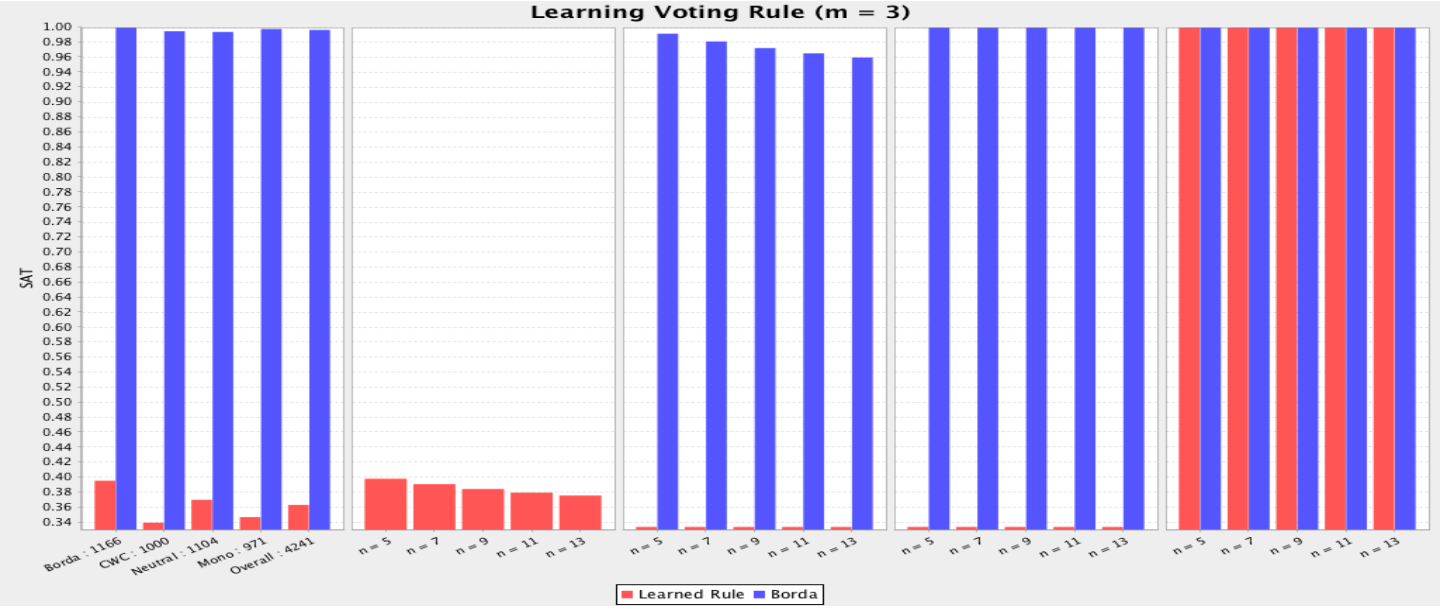

Figure 5: Performance of DL on the positional features

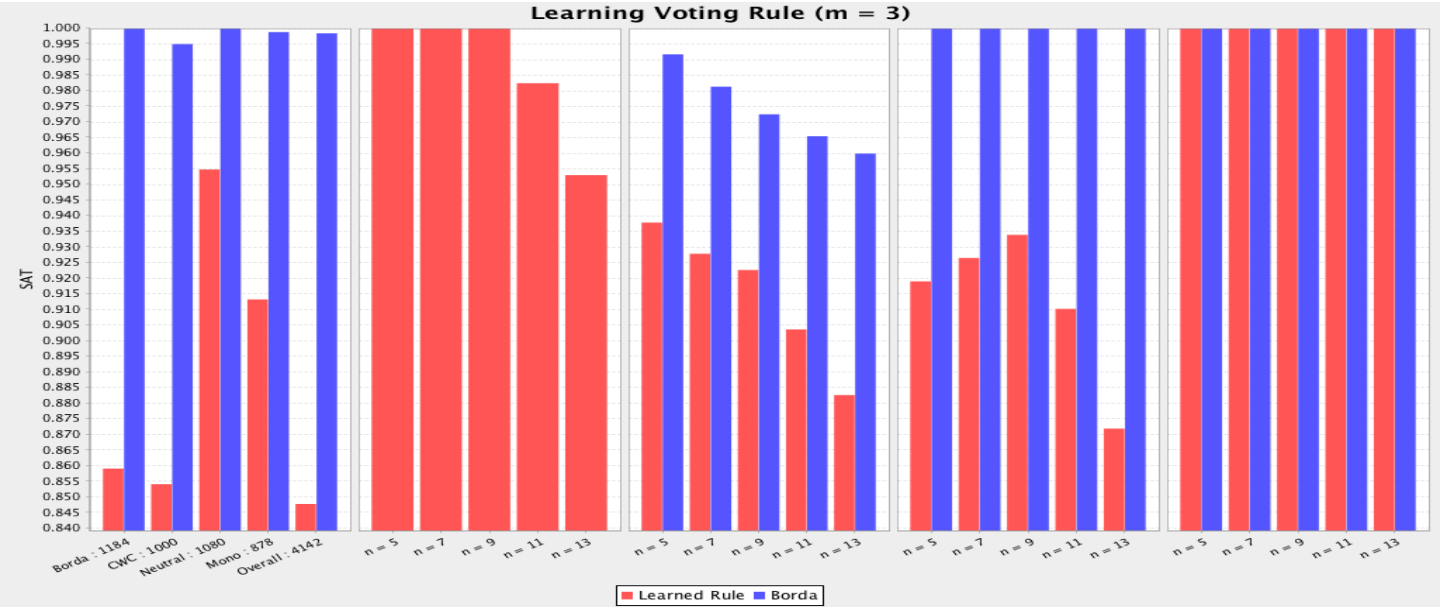

Figure 6: Performance of DL on the pairwise features

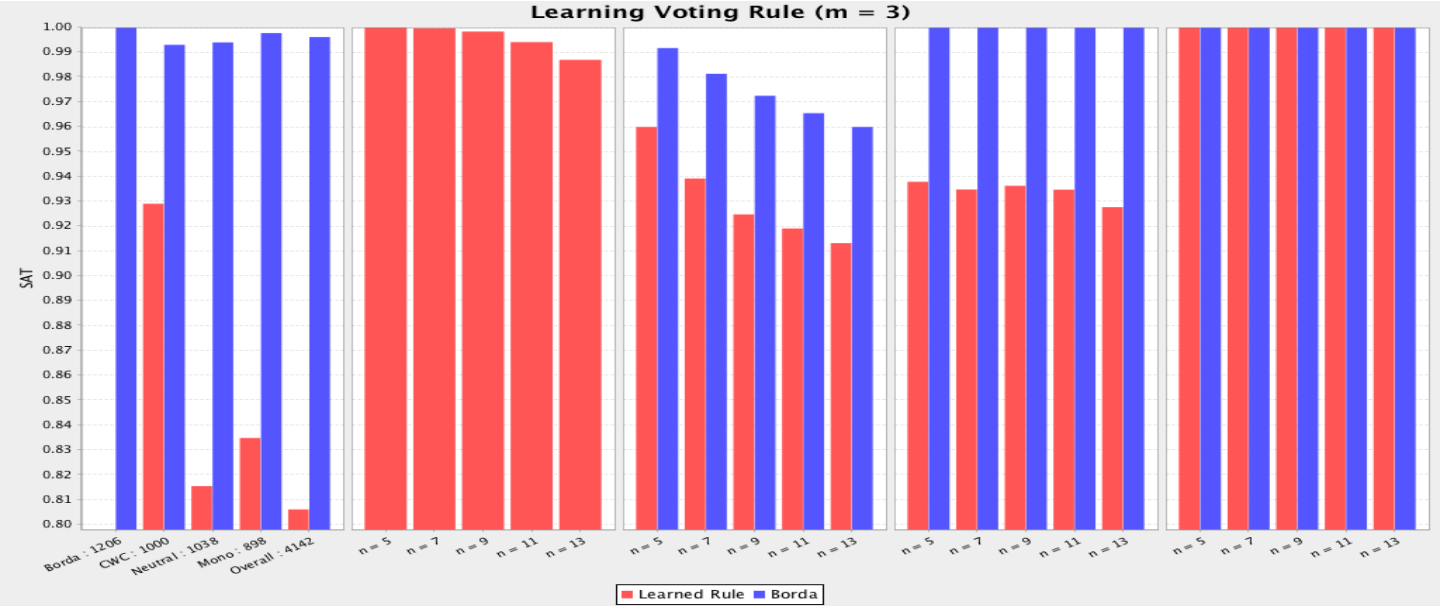

Figure 7: Performance of DL on both positional and pairwise features

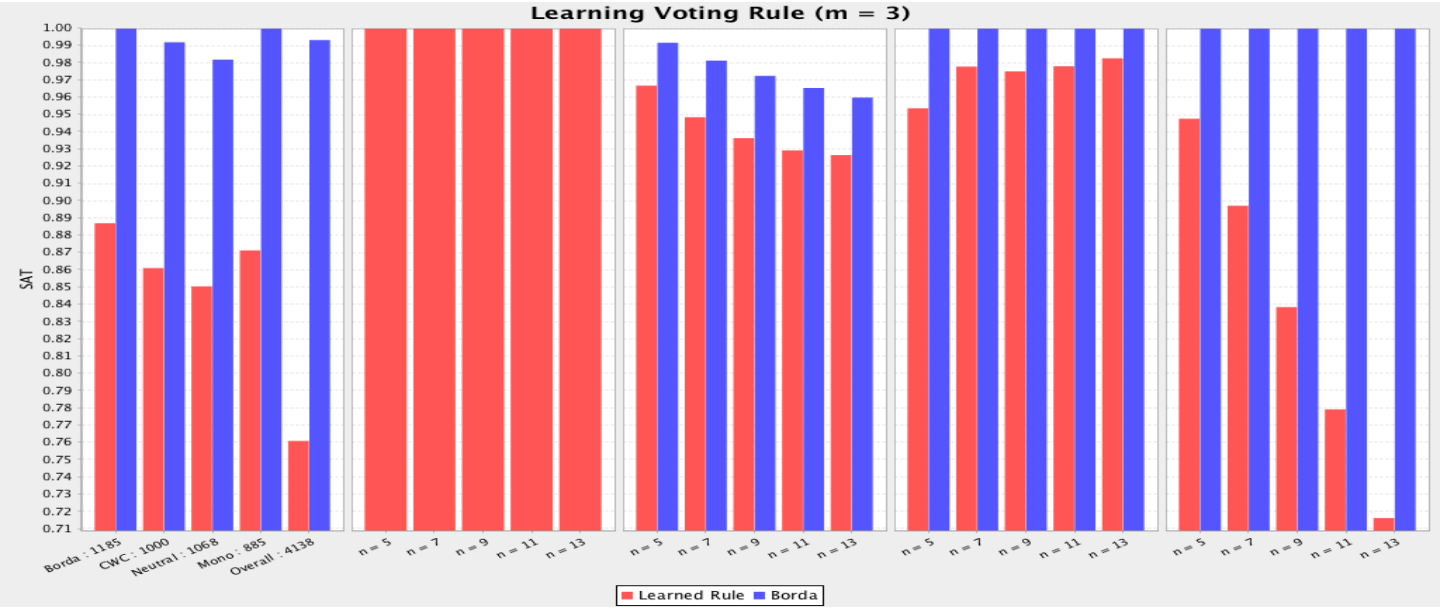

Figure 8: Deep Learning on m!-dimensional Profile Space

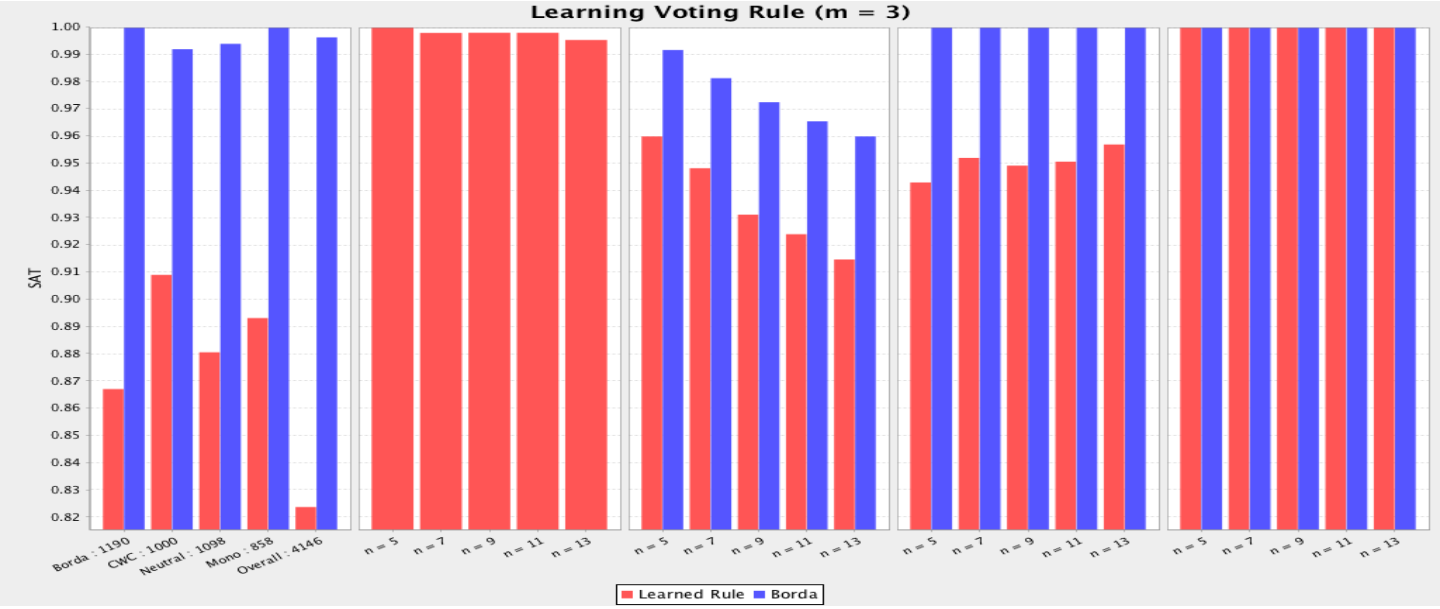

Figure 9: Performance of RF-DL on the pairwise features, $W = [0.5, 0.5]$

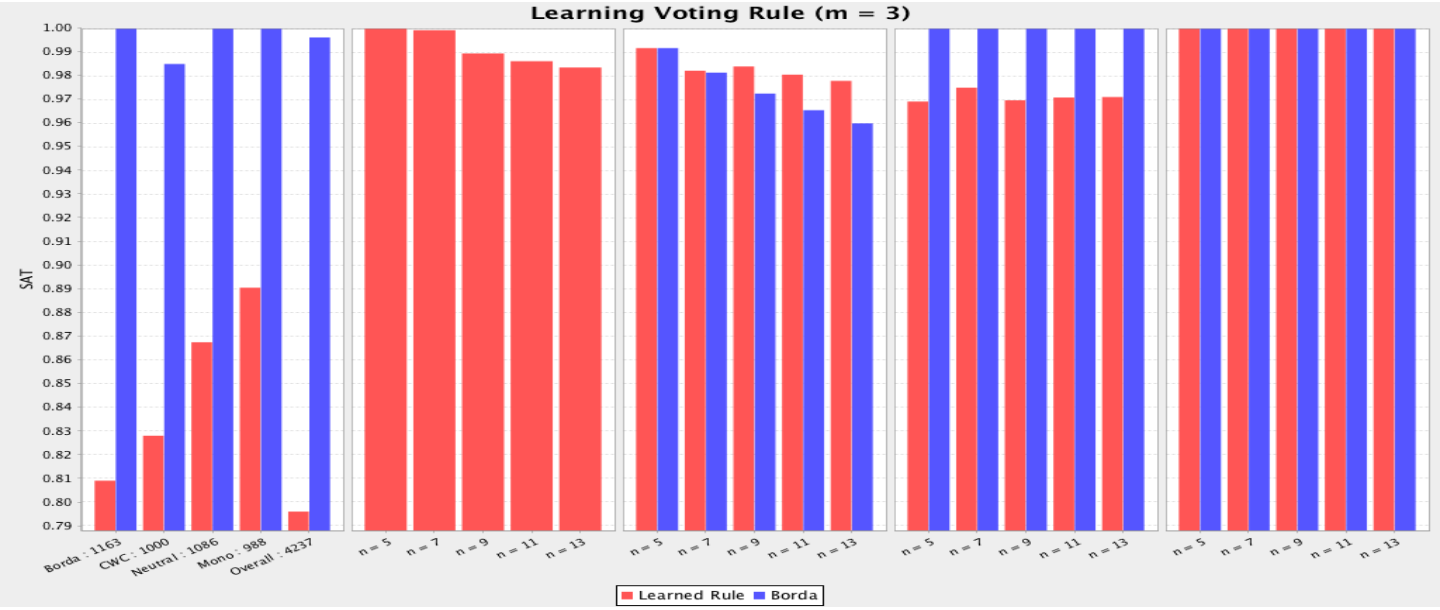

Figure 10: Performance of RF-DL on the pairwise features, $W = [0.5, 0.5]$

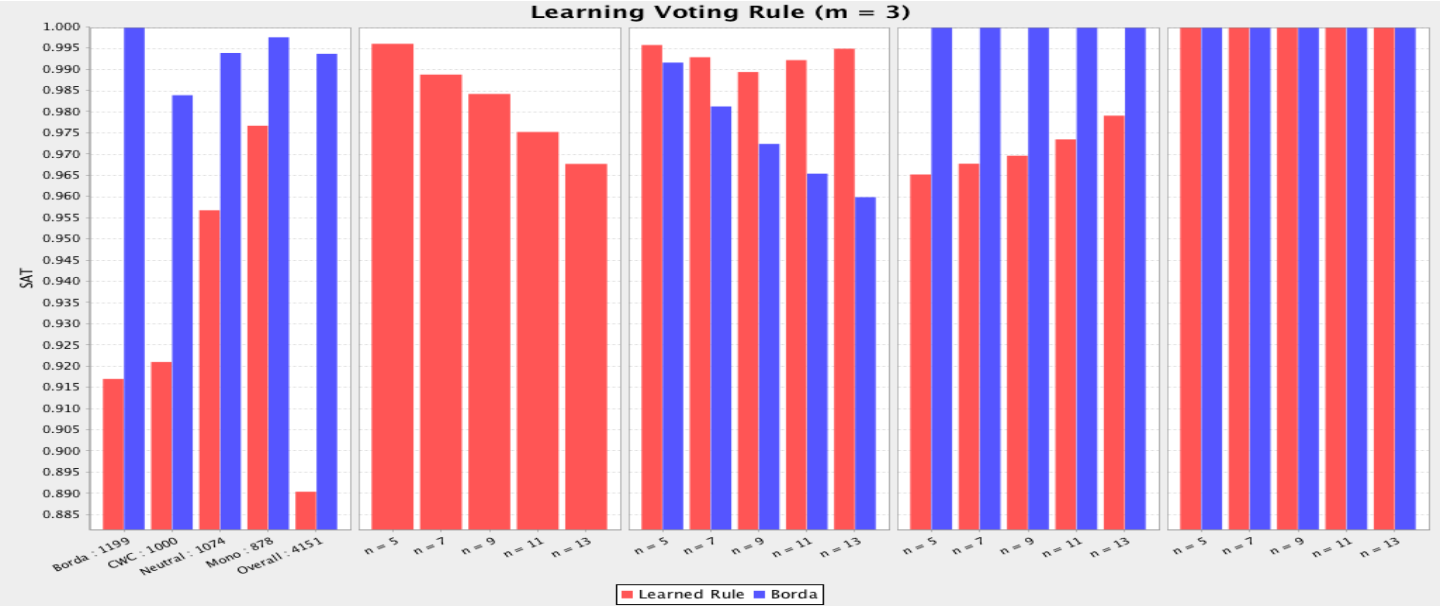

Figure 11: Performance of RF-DL on the pairwise features, $W = [0.4, 0.6]$

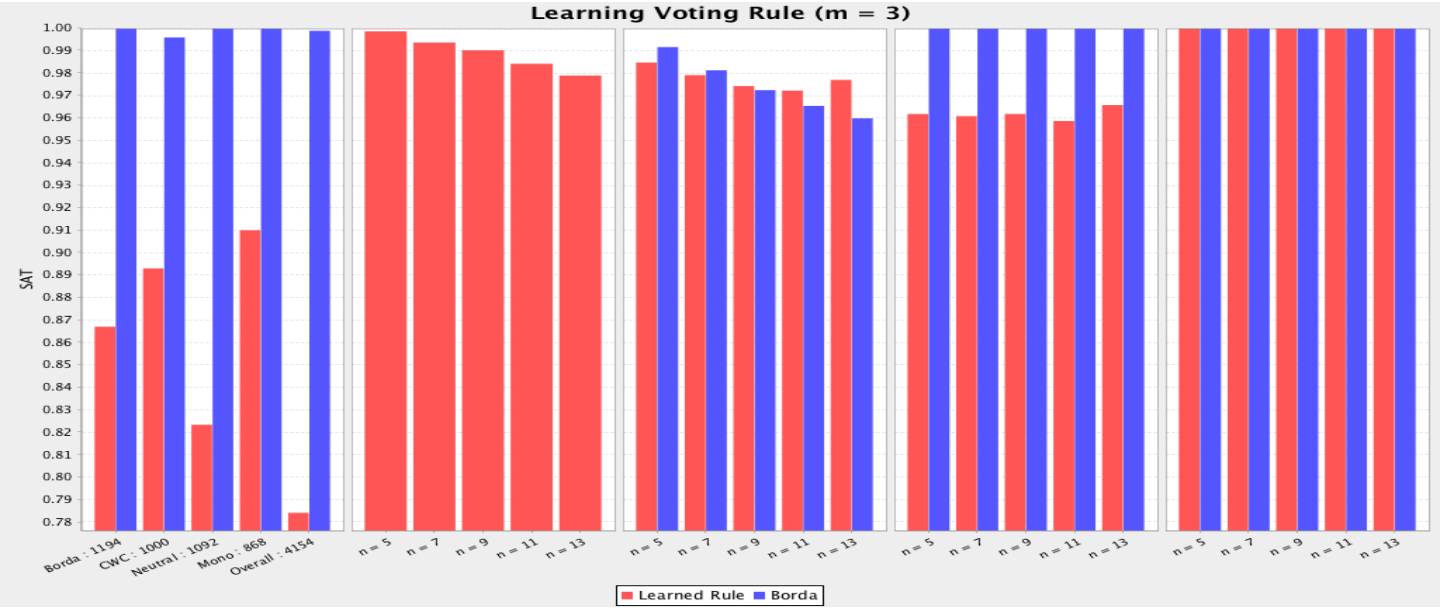

Figure 12: Performance of RF-DL on both positional and pairwise features, $W = [0.5, 0.5]$

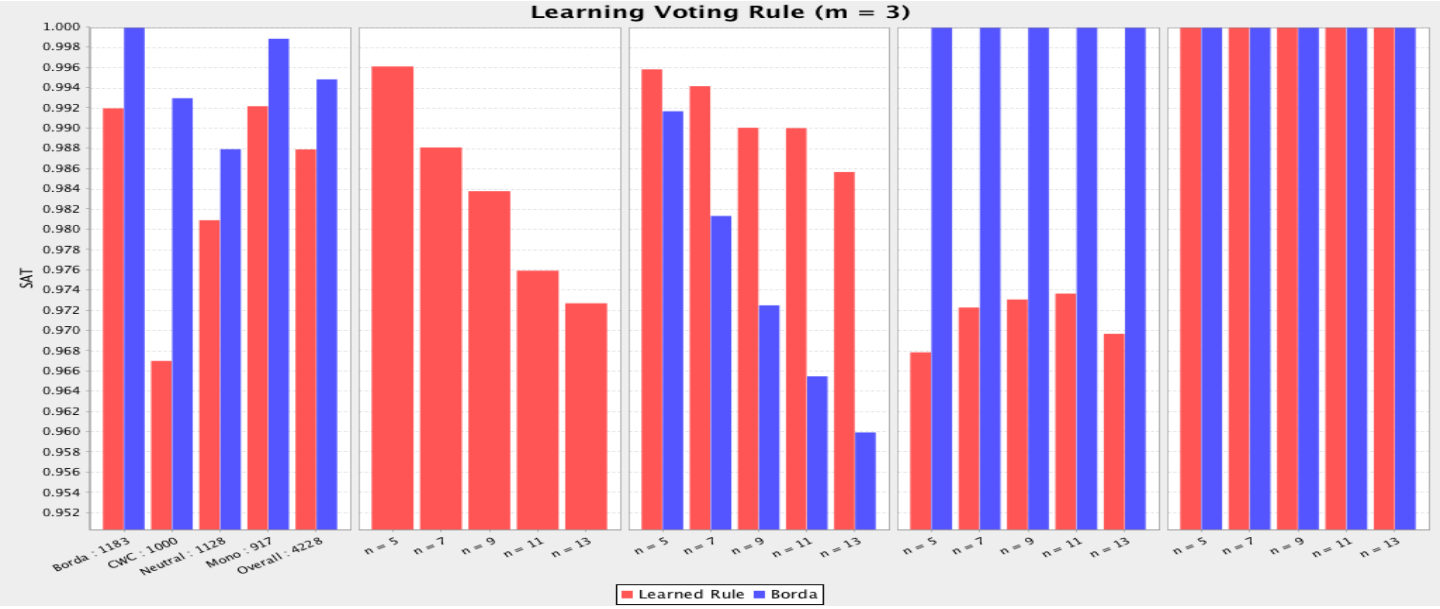

Figure 13: Performance of RF-DL on both positional and pairwise features , $W = [0.6, 0.4]$

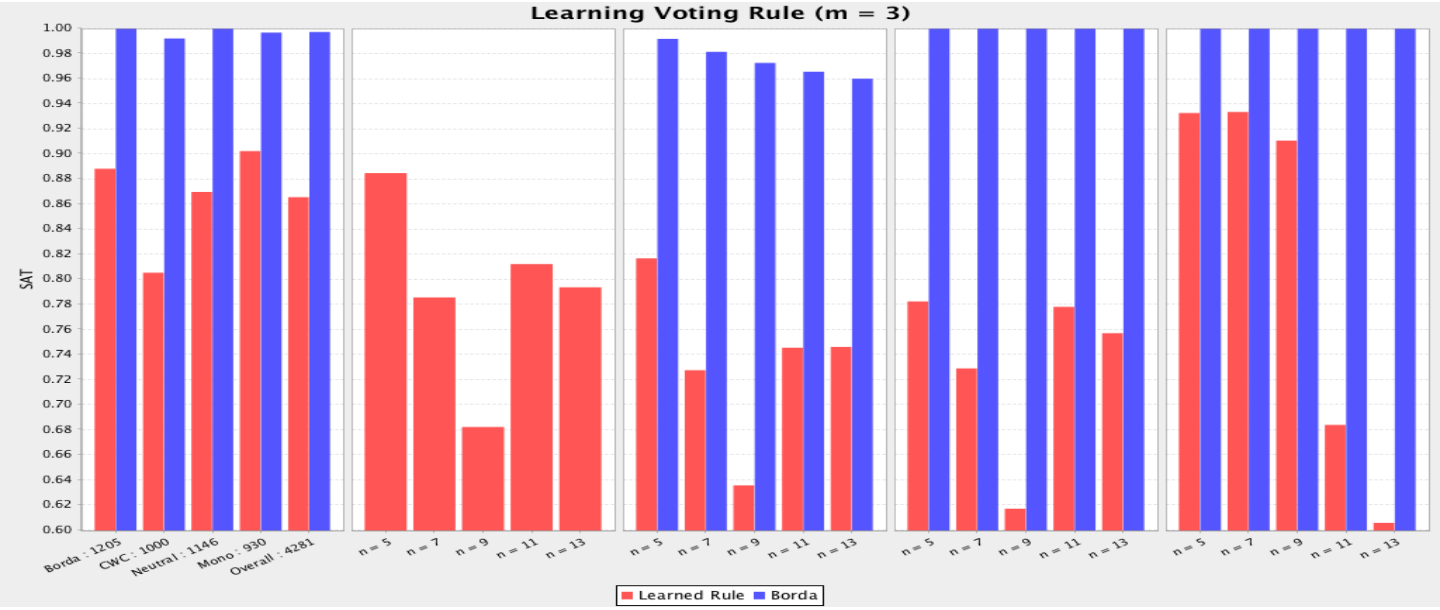

Figure 14: Performance of RF-DL on the $m!$-preference profiles, $W = [0.5, 0.5]$

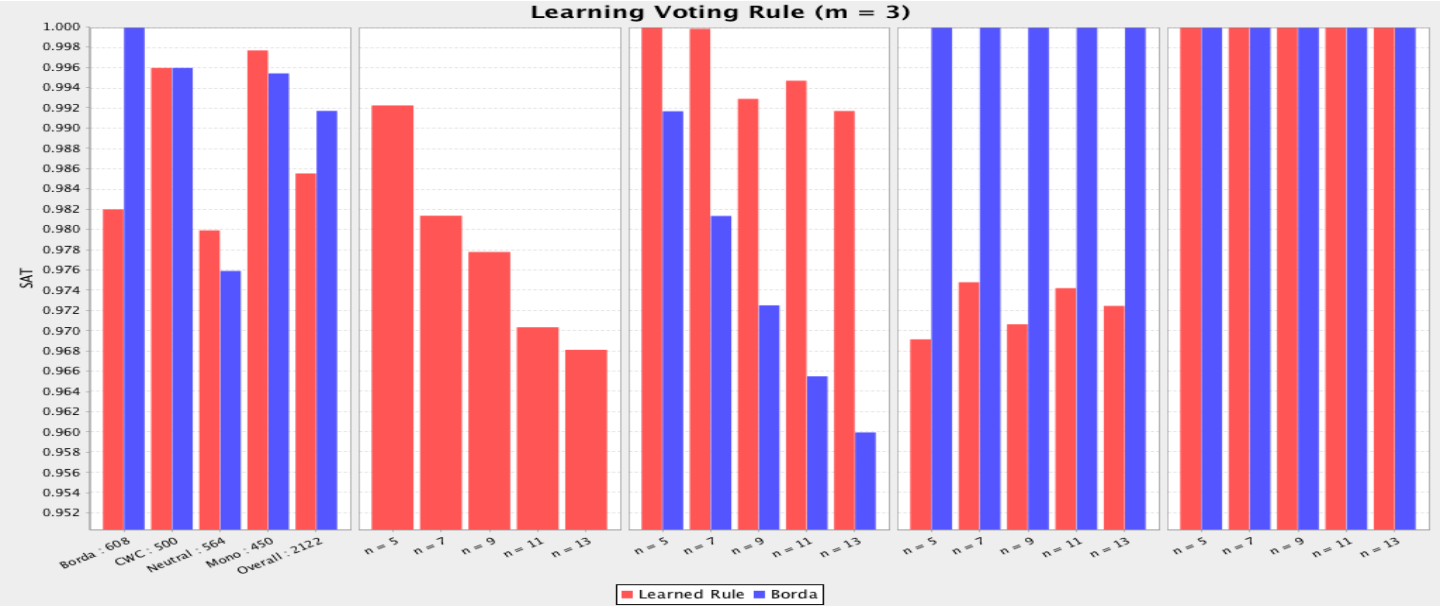

Figure 15: Performance of RF-DL on both positional and pairwise features with less samples, $W = [0.4, 0.6]$

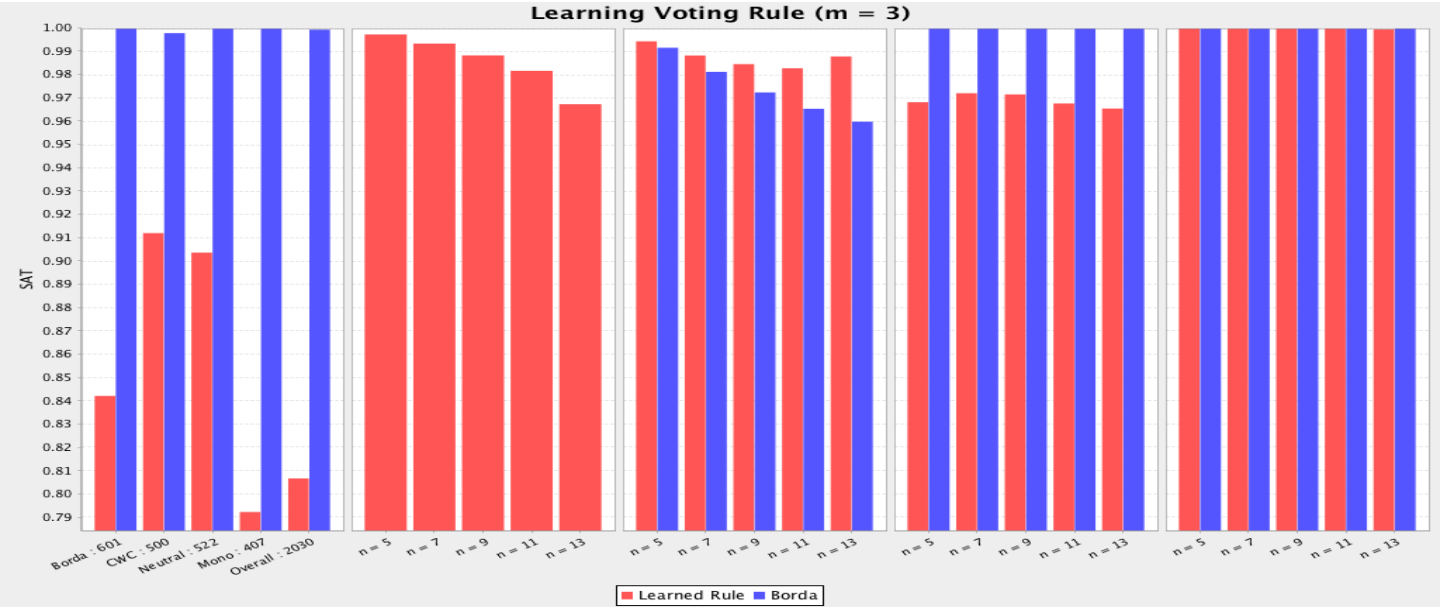

Figure 16: Performance of RF-DL on both positional and pairwise features with less samples, $W = [0.5, 0.5]$