Interpretable Two-Level Boolean Rule Learning for Classification

Guolong Su GUOLONG@MIT.EDU
Massachusetts Institute of Technology, 50 Vassar St., Cambridge, MA 02139 USA

Dennis Wei DWEI@US.IBM.COM
Kush R. Varshney KRVARSHN@US.IBM.COM
Dmitry M. Malioutov DMALIOUTOV@US.IBM.COM
IBM Thomas J. Watson Research Center, 1101 Kitchawan Rd., Yorktown Heights, NY 10598 USA

Abstract

As a contribution to interpretable machine learning research, we develop a novel optimization framework for learning accurate and sparse two-level Boolean rules. We consider rules in both conjunctive normal form (AND-of-ORs) and disjunctive normal form (OR-of-ANDs). A principled objective function is proposed to trade classification accuracy and interpretability, where we use Hamming loss to characterize accuracy and sparsity to characterize interpretability. We propose efficient procedures to optimize these objectives based on linear programming (LP) relaxation, block coordinate descent, and alternating minimization. Experiments show that our new algorithms provide very good tradeoffs between accuracy and interpretability.

1. Introduction

In applications where machine learning is used to aid human decision-making, it is recognized that interpretability of models is an important objective for establishing trust, adoption and safety and for offering the possibility of auditing and debugging (Freitas, 2014; Varshney, 2016; Lima et al., 2009; Letham et al., 2012; Vellido et al., 2012). Interpretable models can be learned directly from data or can result from the approximation of black-box models (Craven & Shavlik, 1996; Baesens et al., 2003).

Boolean rules are considered to be one of the most interpretable classification models (Freitas, 2014). A one-level rule is a conjunctive clause (AND-rule) or disjunctive clause (OR-rule) whereas a two-level rule is a conjunctive normal form (CNF; AND-of-ORs) or disjunctive normal form (DNF; OR-of-ANDs). A mathematical proxy for interpretability of Boolean rules is *sparsity*, i.e. the total number of features used in the rule (Feldman, 2000).

Learning accurate and sparse two-level Boolean rules is a considerable challenge since it is combinatorial (Kearns et al., 1987). Even the simpler problem of learning a one-level rule is NP-hard (Malioutov & Varshney, 2013). Unlike one-level rules, two-level rules can represent any Boolean function of the input features (Fürnkranz et al., 2012); this expressiveness of two-level rules also suggests that they are more challenging to learn than one-level rules. Due to this complexity, most existing solutions focus on heuristic and greedy approaches.

The main contribution of this paper is to introduce a unified optimization framework for learning two-level Boolean rules that achieve good balance between accuracy and interpretability, as measured by the sparsity of the rule. The objective function is a weighted combination of (a) classification errors quantified by Hamming distance between the current rule and the closest rule that correctly classifies a sample and (b) sparsity. Based on this formulation, block coordinate descent and alternating minimization algorithms are developed, both of which use an LP relaxation approach. Experiments show that two-level rules can have considerably higher accuracy than one-level rules and may show improvements over cutting edge approaches.

The two-level Boolean rules in this work are examples of sparse decision rule lists (Rivest, 1987), which have been extensively studied in various fields. A number of strategies have been proposed (Fürnkranz et al., 2012): covering (Rivest, 1987; Clark & Niblett, 1989; Cohen, 1995; Fürnkranz, 1999), bottom-up (Salzberg, 1991; Domingos, 1996), multi-phase (Liu et al., 1998; Knobbe et al., 2008), and the distillation of trained decision trees into decision lists (Quinlan, 1987). Unlike our proposed approach, the above strategies lack a single, principled objective function

2016 ICML Workshop on Human Interpretability in Machine Learning (WHI 2016), New York, NY, USA.

to drive rule learning. Moreover, they employ heuristics that leave room for improvements on both accuracy and rule simplicity. In addition to the symbolic approaches above, Bayesian approaches in Letham et al. (2012) and Wang et al. (2015) apply approximate inference algorithms to produce posterior distributions over decision lists; however, the assignment of prior and likelihood may not always be clear, and certain approximate inference algorithms may have high computational cost.

There has been some prior work on optimization-based formulations for rule learning, the most relevant being Malioutov & Varshney (2013), where an LP framework is proposed to learn one-level rules from which set covering and boosting are used to construct two-level rules. Although we apply this clause learning method as a component in our algorithms, our work has significant differences from Malioutov & Varshney (2013). As discussed earlier, two-level rules are significantly more expressive and much more challenging to learn than a one-level rule. In addition, the greedy style of the set covering method leaves room for improvement and the weighted combination of clauses in boosted classifiers reduces interpretability. Another work on DNF learning (Wang & Rudin, 2015) provides a mixed integer program (MIP) formulation named OOA and a different heuristic formulation OOAx. The MIP in OOA is similar to our formulation with a different cost (0-1 error) but lacks an LP relaxation. OOAx is similar to the heuristic multi-phase strategy above.

2. Problem Formulation

We consider supervised binary classification given a training dataset of n samples, where each sample has a label $y_i \in \{0,1\}$ and d binary features[1] $a_{i,j} \in \{0,1\}$ $(1 \le j \le d)$. The goal is to learn a classifier $\hat{y}(\cdot)$ in CNF (AND-of-ORs) that can generalize well from the training dataset.[2] In the lower level of the rule, each of R clauses is formed by the disjunction of a selected subset of input features; in the upper level, the predictor is obtained as the conjunction of all clauses. Letting the decision variables $w_{j,r} \in \{0,1\}$ represent whether to include the j^{th} feature in the r^{th} clause, the clause and predictor outputs are given by

$$\hat{v}_{i,r} = \bigvee_{j=1}^{d} (a_{i,j} w_{j,r}), \text{ for } 1 \le i \le n,\ 1 \le r \le R. \quad (1)$$

$$\hat{y}_i = \bigwedge_{r=1}^{R} \hat{v}_{i,r}, \text{ for } 1 \le i \le n. \quad (2)$$

[1] We assume the negation of each feature is included as another input feature.

[2] The presentation focuses on CNF rules, but the proposed algorithms apply equally to learning DNF rules using De Morgan's laws.

To mitigate the need for careful specification of the number of clauses R, we allow clauses to be "disabled" by padding the input feature matrix with a trivial "always true" feature $a_{i,0} = 1$ for all i, with corresponding decision variables $w_{0,r}$ for all clauses. If $w_{0,r} = 1$, then the r^{th} clause has output 1 and thus drops out of the upper-level conjunction in a CNF rule. The sparsity cost for $w_{0,r}$, i.e. for disabling a clause, can be lower than other variables or even zero.

In learning Boolean rules, it is desirable to use a finer-grained measure of accuracy than the usual 0-1 loss to distinguish between degrees of incorrectness and indicate where corrections are needed. Herein we propose measuring the accuracy of a rule on a single sample in terms of the minimal Hamming distance from the rule to an *ideal* rule, i.e. one that correctly classifies the sample. The Hamming distance between two CNF rules is the number of $w_{j,r}$ that are different in the two rules. Thus the minimal Hamming distance represents the smallest number of modifications (i.e. negations) needed to correct a misclassification.

For mathematical formulation, we introduce *ideal clause outputs* $v_{i,r}$ with $1 \le i \le n$ and $1 \le r \le R$ to represent a CNF rule that correctly classifies the i^{th} sample. The values of $v_{i,r}$ are always consistent with the ground truth labels, i.e. $y_i = \bigwedge_{r=1}^{R} v_{i,r}$ for all $1 \le i \le n$. We let $v_{i,r}$ have a ternary alphabet $\{0, 1, \text{DC}\}$, where $v_{i,r} = \text{DC}$ means that we "don't care" about the value of $v_{i,r}$. With this setup, if $y_i = 1$, then $v_{i,r} = 1$ for all $1 \le r \le R$; if $y_i = 0$, then $v_{i,r_0} = 0$ for at least one value of r_0, and we can have $v_{i,r} = \text{DC}$ for all $r \ne r_0$. In implementation, $v_{i,r} = \text{DC}$ implies the removal of the i^{th} sample in the training or updating for the r^{th} clause, which leads to a different training subset for each clause.

For a given $v_{i,r}$, the minimal Hamming distance between the r^{th} clauses only of a CNF rule and an ideal rule can be derived as follows. If $v_{i,r} = 1$, at most one positive feature needs to be included to produce the desired output, so the minimal Hamming distance is given by $\max\{0, 1 - \sum_{j=1}^{d} a_{i,j} w_{j,r}\}$. If $v_{i,r} = 0$, any $w_{j,r}$ with $a_{i,j} w_{j,r} = 1$ needs to be negated to be correct, resulting in a minimal Hamming distance of $\sum_{j=1}^{d} a_{i,j} w_{j,r}$. Summing over i, r and defining the sparsity cost as the sum of the numbers of features used in each clause, the problem is formulated as

$$\min_{w_{j,r},\, v_{i,r}} \sum_{i=1}^{n} \sum_{r=1}^{R} \left[\mathbb{1}_{v_{i,r}=1} \cdot \max\left\{0, \left(1 - \sum_{j=1}^{d} a_{i,j} w_{j,r}\right)\right\} + \mathbb{1}_{v_{i,r}=0} \cdot \sum_{j=1}^{d} a_{i,j} w_{j,r} \right] + \theta \cdot \sum_{r=1}^{R} \sum_{j=1}^{d} w_{j,r} \quad (3)$$

$$\text{s.t.} \bigwedge_{r=1}^{R} v_{i,r} = y_i,\ \forall i, \quad (4)$$

$$v_{i,r} \in \{0, 1, \text{DC}\},\ w_{j,r} \in \{0,1\},\ \forall i, j, r.$$

The ideal clause output constraint (4) requires that $v_{i,r} = 1$ for all r if $y_i = 1$, as noted above. For $y_i = 0$, $v_{i,r} = 0$ needs to hold for at least one value of r while all other $v_{i,r}$ can be DC. The Hamming distance is minimized by setting

$$v_{i,r_0} = 0, \text{ where } r_0 = \arg\min_{1 \le r \le R} \left(\sum_{j=1}^{d} a_{i,j} w_{j,r} \right). \quad (5)$$

The binary variables $w_{j,r}$ can be further relaxed to $0 \le w_{j,r} \le 1$. However, the resulting continuous relaxation is generally non-convex for $R > 1$. Additional simplifications are proposed in Section 3 to make the continuous relaxations more efficiently solvable.

Lastly, it can be seen that letting $R = 1$ in formulation (3) recovers the formulation for one-level rule learning in Malioutov & Varshney (2013).

3. Optimization Approaches

This section proposes a block coordinate descent algorithm and an alternating minimization algorithm to solve the regularized Hamming loss minimization in (3).

3.1. Block Coordinate Descent Algorithm

This algorithm considers the decision variables in a single clause ($w_{j,r}$ with a fixed r) as a block of coordinates, and performs block coordinate descent to minimize the Hamming distance objective function in (3). Each iteration updates a single clause with all the other $(R-1)$ clauses fixed, using the one-level rule learning algorithm in Malioutov & Varshney (2013). We denote r_0 as the clause to be updated.

The optimization of (3) even with $(R-1)$ clauses fixed still involves a joint minimization over w_{j,r_0} and the ideal clause outputs $v_{i,r}$ for $y_i = 0$ ($v_{i,r} = 1$ for $y_i = 1$ are fixed), so the exact solution could still be challenging. To simplify, we fix the values of $v_{i,r}$ for $y_i = 0$ and $r \neq r_0$ to the actual clause outputs $\hat{v}_{i,r}$ in (1) with the current $w_{j,r}$ ($r \neq r_0$). Now we assign v_{i,r_0} for $y_i = 0$: if there exists $v_{i,r} = \hat{v}_{i,r} = 0$ with $r \neq r_0$, then this sample is guaranteed to be correctly classified and we can assign $v_{i,r_0} = \text{DC}$ to minimize the objective in (3); in contrast, if $\hat{v}_{i,r} = 1$ holds for all $r \neq r_0$, then the constraint (4) requires $v_{i,r_0} = 0$.

This derivation leads to the updating process as follows. To update the r_0^{th} clause, we remove all samples that have label $y_i = 0$ and are already predicted as 0 by at least one of the other $(R-1)$ clauses, and then update the r_0^{th} clause with the remaining samples using the one-level rule learning algorithm.

There are different choices of which clause to update in an iteration. For example, we can update clauses cyclically or randomly, or we can try the update for each clause and then greedily choose the one with the minimum cost. The greedy update is used in our experiments.

The initialization of $w_{j,r}$ in this algorithm also has different choices. For example, one option is the set covering method, as is used in our experiments. Random or all-zero initialization can also be used.

3.2. Alternating Minimization Algorithm

This algorithm alternately minimizes with respect to the decision variables $w_{j,r}$ and the ideal clause outputs $v_{i,r}$ in (3). Each iteration has two steps: update $v_{i,r}$ with the current $w_{j,r}$, and update $w_{j,r}$ with the new $v_{i,r}$. The latter step is simpler and will be first discussed.

With fixed values of $v_{i,r}$, the minimization over $w_{j,r}$ is relatively straight-forward: the objective in (3) is separated into R terms, each of which depends only on a single clause $w_{j,r}$ with a fixed r. Thus, all clauses are decoupled in the minimization over $w_{j,r}$, and the problem becomes parallel learning of R one-level clauses. Explicitly, the update of the r^{th} clause removes samples with $v_{i,r} = \text{DC}$ and then uses the one-level rule learning algorithm.

The update over $v_{i,r}$ with fixed $w_{j,r}$ follows the discussion in Section 2: for positive samples $y_i = 1$, $v_{i,r} = 1$, and for the negative samples $y_i = 0$, $v_{i,r_0} = 0$ for r_0 defined in (5) and $v_{i,r} = \text{DC}$ for $r \neq r_0$. For negative samples with a "tie", i.e. non-unique r_0 in (5), tie breaking is achieved by a "clustering" approach. First, for each clause $1 \le r_0 \le R$, we compute its cluster center in the feature space by taking the average of $a_{i,j}$ (for each j) over samples i for which r_0 is minimal in (5) (including ties). Then, each sample with a tie is assigned to the clause with the closest cluster center in ℓ_1-norm among all minimal r_0 in (5).

Similar to the block coordinate descent algorithm, various options exist for initializing $w_{j,r}$ in this algorithm. The set covering approach is used in our experiments.

4. Numerical Evaluation

This section evaluates the algorithms with UCI repository datasets (Lichman, 2013). To facilitate comparison with the most relevant prior work (Malioutov & Varshney, 2013), we use all 8 datasets in that work. Each continuous valued feature is converted to binary using 10 quantile thresholds. In addition, we use 2 large datasets: MAGIC gamma telescope (MAGIC) and Musk version 2 (Musk).

The goal is to learn a DNF rule (OR-of-ANDs) from each dataset. We use stratified 10-fold cross validation and then average the error rates. All LPs are solved by CPLEX version 12 (IBM). The sparsity parameter θ is tuned between 10^{-4} and 50 using a second cross validation

Table 1. 10-fold Average Test Error Rates (unit: %). Standard Error of the Mean is Shown in Parentheses.

DATASET	BCD	AM	OCRL	SC	DLIST	C5.0	CART	RIPPER
ILPD	28.6(0.2)	28.6(0.2)	28.6(0.2)	28.6(0.2)	36.5(1.4)	30.5(2.0)	32.8(1.3)	31.9(1.0)
IONOS	9.4(1.1)	11.4(1.1)	9.7(1.5)	10.5(1.3)	19.9(2.3)	7.4(2.1)	10.8(1.2)	10.0(1.5)
LIVER	37.1(3.2)	39.1(2.5)	45.8(2.2)	41.7(2.8)	45.2(2.6)	36.5(2.4)	37.1(2.5)	35.4(2.2)
MAGIC	17.4(0.2)	17.0(0.1)	23.6(0.3)	19.7(0.2)	18.5(0.4)	14.1(0.2)	17.8(0.3)	15.1(0.3)
MUSK	7.5(0.3)	3.6(0.6)	8.4(0.1)	8.1(0.3)	13.6(0.5)	3.1(0.4)	3.3(0.3)	3.7(0.2)
PARKIN	12.8(2.2)	15.9(2.9)	16.4(2.1)	14.9(1.9)	25.1(3.3)	16.4(2.7)	13.9(2.9)	10.7(1.8)
PIMA	26.8(1.7)	23.8(2.0)	27.2(1.5)	27.9(1.5)	31.4(1.6)	24.9(1.7)	27.3(1.5)	24.9(1.1)
SONAR	29.8(3.0)	25.5(2.4)	34.6(2.7)	28.8(2.9)	38.5(3.6)	25.0(4.2)	31.7(3.5)	25.5(3.1)
TRANS	23.8(0.1)	23.8(0.1)	23.8(0.1)	23.8(0.1)	35.4(2.4)	21.7(1.2)	25.4(1.7)	21.5(0.8)
WDBC	6.2(1.2)	6.5(0.9)	9.3(2.0)	8.8(2.0)	9.7(0.8)	6.5(1.1)	8.4(1.0)	7.4(1.2)
RANKING	3.1	3.2	5.5	4.9	7.7	2.6	5.0	2.8

Table 2. 10-fold Average Numbers of Features

DATASET	BCD	AM	SC	DLIST	C5.0	RIPPER
ILPD	0.0	0.0	0.0	5.4	45.5	7.0
IONOS	12.4	12.9	11.1	7.7	13.6	6.0
LIVER	9.3	7.7	5.2	2.2	46.6	4.0
MAGIC	11.4	22.3	2.0	14.7	366.7	110.0
MUSK	26.5	63.7	18.3	15.9	155.1	92.0
PARKIN	8.2	12.6	3.2	2.1	16.6	6.0
PIMA	2.2	2.0	2.4	8.6	38.2	5.0
SONAR	14.2	23.6	9.0	1.9	27.3	8.0
TRANS	0.0	0.0	0.0	3.8	6.7	5.0
WDBC	13.6	11.9	8.7	4.0	15.8	6.0
RANKING	3.1	3.4	2.2	2.3	6.0	3.4

within the training partition.

Algorithms in comparison and their abbreviations are: block coordinate descent (BCD), alternating minimization (AM), one-level conjunctive rule learning (OCRL, equivalent to setting $R = 1$ for BCD or AM) and set covering (SC), the last two from Malioutov & Varshney (2013), decision list in IBM SPSS (DList), decision trees (C5.0: C5.0 with rule set option in IBM SPSS, CART: classification and regression trees algorithm in Matlab's classregtree function), and RIPPER from Cohen (1995). We set the maximum number of clauses $R = 5$ for the BCD, AM, and SC algorithms, and set the maximum number of iterations in BCD and AM as 100.

We first show the test error rates and the sparsity of the rules. The mean test error rates and the standard error of the mean are listed in Table 1. Due to space constraints, we refer the reader to Malioutov & Varshney (2013) for results from other classifiers that are not interpretable; the accuracy of our algorithms is generally quite competitive with them. Table 2 provides the 10-fold average of the sparsity of the learned rules as a measure for interpretability. No features are counted if a clause is disabled. The last rows in these tables show the averaged ranking of each algorithm on each dataset.

Table 1 shows that two-level rules obtained by our algorithms (BCD and AM) are more accurate than the one-level rules from OCRL for almost all datasets, which demonstrates the expressiveness of two-level rules. Among optimization-based two-level rule learning approaches, BCD and AM generally have superior accuracy to SC. All these approaches substantially outperform DList in terms of accuracy on all datasets. Compared with C5.0, BCD and AM obtain rules with much higher interpretability (many fewer features) and quite competitive accuracy. Compared with CART, BCD has higher or equal accuracy on all datasets except for Musk, and AM is also superior overall. RIPPER appears to be slightly stronger than BCD and AM for the 8 datasets from Malioutov & Varshney (2013). However, on the two larger datasets (MAGIC and Musk), RIPPER selects a rather large number of features, and further study on large datasets is needed to clarify the advantages and disadvantages of the algorithms. In addition, AM achieves the highest accuracy on Pima, and BCD obtains the highest accuracy on WDBC.

Below is an example of a learned rule that predicts Parkinson's disease from voice features. (It is consistent with known low frequency and volume change reduction in the voices of Parkinson's patients (Ramig et al., 2004).)

IF 1. voice fractal scaling exponent > -6.7; OR
2. max vocal fundamental frequency < 236.4 Hz; AND
min vocal fundamental frequency < 181.0 Hz; AND
shimmer:DDA < 0.0361; AND
recurrence period density entropy < 0.480;
THEN this person has Parkinson's.

5. Conclusion

Motivated by the need for interpretable classification models, this paper has provided an optimization-based formulation for two-level Boolean rule learning. These complement the more heuristic strategies in the literature on two-level Boolean rules. Numerical results show that two-level Boolean rules typically have considerably lower error rate than one-level rules and provide very good tradeoffs between accuracy and interpretability with improvements over state-of-the-art approaches.

Acknowledgment

The authors thank V. S. Iyengar, A. Mojsilović, K. N. Ramamurthy, and E. van den Berg for conversations and support.

References

IBM ILOG CPLEX optimization studio. http://www-03.ibm.com/software/products/en/ibmilogcpleoptistud.

Baesens, B., Setiono, R., Mues, C., and Vanthienen, J. Using neural network rule extraction and decision tables for credit-risk evaluation. *Manag. Sci.*, 49(3):312–329, 2003.

Clark, P. and Niblett, T. The CN2 induction algorithm. *Mach. Learn.*, 3(4):261–283, 1989.

Cohen, W. W. Fast effective rule induction. In *Proc. Int. Conf. Mach. Learn.*, pp. 115–123, 1995.

Craven, M. W. and Shavlik, J. W. Extracting tree-structured representations of trained networks. *Adv. Neural Inf. Process. Syst.*, pp. 24–30, 1996.

Domingos, P. Unifying instance-based and rule-based induction. *Mach. Learn.*, 24(2):141–168, 1996.

Feldman, J. Minimization of Boolean complexity in human concept learning. *Nature*, 407(6804):630–633, 2000.

Freitas, A. A. Comprehensible classification models – a position paper. *ACM SIGKDD Explor.*, 15(1):1–10, 2014.

Fürnkranz, J. Separate-and-conquer rule learning. *Artif. Intell. Rev.*, 13(1):3–54, 1999.

Fürnkranz, J., Gamberger, D., and Lavrač, N. *Foundations of rule learning*. Springer Science & Business Media, 2012.

Kearns, M., Li, M., Pitt, L., and Valiant, L. On the learnability of Boolean formulae. In *Proc. Annu. ACM Symp. on Theory of Comput.*, pp. 285–295, 1987.

Knobbe, A., Crémilleux, B., Fürnkranz, J., and Scholz, M. From local patterns to global models: The LeGo approach to data mining. In *Proc. ECML PKDD Workshop (LeGo-08)*, pp. 1–16, 2008.

Letham, B., Rudin, C., McCormick, T. H., and Madigan, D. Building interpretable classifiers with rules using Bayesian analysis. *Department of Stat. Tech. Report tr609, Univ. of Washington*, 2012.

Lichman, M. UCI machine learning repository. http://archive.ics.uci.edu/ml, Univ. of Calif., Irvine, School of Information and Computer Sciences, 2013.

Lima, E., Mues, C., and Baesens, B. Domain knowledge integration in data mining using decision tables: case studies in churn prediction. *J. Oper. Res. Soc.*, 60(8): 1096–1106, 2009.

Liu, B., Hsu, W., and Ma, Y. Integrating classification and association rule mining. In *Proc. Int. Conf. Knowl. Discov. Data Min.*, pp. 80–86, 1998.

Malioutov, D. M. and Varshney, K. R. Exact rule learning via Boolean compressed sensing. In *Proc. Int. Conf. Mach. Learn.*, pp. 765–773, 2013.

Quinlan, J. R. Simplifying decision trees. *Int. J. Man-Mach. Studies*, 27(3):221–234, 1987.

Ramig, L. O., Fox, C., and Sapir, S. Parkinson's disease: speech and voice disorders and their treatment with the lee silverman voice treatment. In *Seminars in Speech and Language*, volume 25, pp. 169–180, 2004.

Rivest, R. L. Learning decision lists. *Mach. Learn.*, 2(3): 229–246, 1987.

Salzberg, S. A nearest hyperrectangle learning method. *Mach. Learn.*, 6(3):251–276, 1991.

Varshney, K. R. Engineering safety in machine learning. In *Proc. Inf. Theory Appl. Workshop*, 2016.

Vellido, A., Martín-Guerrero, J. D., and Lisboa, P. J.G. Making machine learning models interpretable. In *Proc. Eur. Sym. Artif. Neural Networ. Comput. Intell. Mach. Learn.*, pp. 163–172, 2012.

Wang, T. and Rudin, C. Learning optimized Or's of And's. *arXiv preprint arXiv:1511.02210*, 2015.

Wang, T., Rudin, C., Doshi-Velez, F., Liu, Y., Klampfl, E., and MacNeille, P. Bayesian Or's of And's for interpretable classification with application to context aware recommender systems. Technical report, MIT, 2015. Submitted.