Sparse Logistic Regression with Logical Features

Yuan Zou(✉) and Teemu Roos

Helsinki Institute for Information Technology HIIT,
Gustaf Hällströmin katu 2b, 00014 Helsinki, Finland
{yuan.zou,teemu.roos}@cs.helsinki.fi
http://www.hiit.fi/cosco/promo

Abstract. Modeling interactions in regression models poses both computational as well as statistical challenges: the computational resources and the amount of data required to solve them increases sharply with the size of the problem. We focus on logistic regression with categorical variables and propose a method for learning dependencies that are expressed as general Boolean formulas. The computational and statistical challenges are solved by applying a technique called transformed Lasso, which involves a matrix transformation of the original covariates. We compare the method to an earlier related method, LogicReg, and show that our method scales better in terms of the number of covariates as well as the order and complexity of the interactions.

Keywords: Feature selection · Logistic regression · Lasso

1 Introduction

A basic logistic regression model includes individual effects of feature variables (a.k.a. regressors or covariates) on the probability of a response event. In addition to the individual effects, interactions between the feature variables are important in a wide range of applications. Examples include identifying important single nucleotide polymorphism (SNPs) in genome-wide association studies [6], pathway analysis of gene-expression or metabolomic data [14], regulatory motif identification [7], and association studies of gene-gene interactions [16].

Pairwise and higher order interactions between the features can be modelled by explicitly including the interactions as feature variables. In other words, a kth order interaction can be modelled by merging a subset of k variables into a new variable whose domain becomes the Cartesian product of the domans of the merged variables, and including indicator variables for each of the values in the new domain. High order interactions pose statistical and computational challenges since the number of interaction terms grows exponentially with the order, k. Different techniques for dealing with these challenges have been proposed. These include, for instance, forward and backward selection (see e.g. [5]) and more recently, Lasso [15].

We consider situations where the interactions can be expressed as Boolean formulas, each of which is composed of a subset of the original feature variables connected by logical operations such as AND, OR, and XOR (exclusive or). For example, if we have a vector of m binary variables $\boldsymbol{x} = \{x_1, x_2, \ldots, x_m\}$, the model may involve terms such as "x_1 and x_2, or x_3" or "x_4 and x_5 and not x_6". We define the model formally as a generalized linear model

$$g(E(y)) = \beta_0 + \sum_{i=1}^{t} \beta_i L_i(\boldsymbol{x}), \tag{1}$$

where y is a binary response variable, $E(y) = \Pr[y = 1 \mid \boldsymbol{x}]$ denotes the expectation of y conditional on the regressors $\boldsymbol{x}$, g is a link function, $\beta_0, \ldots, \beta_t$ are regression coefficients, and $L_1, \ldots, L_t$ are Boolean functions of $\boldsymbol{x}$. The right-hand side of Eq. (1) is called a *linear predictor*. Depending on how we define the link function g, this framework includes a range of different model families, such as linear regression, logistic regression, Poisson regression, etc. In this study, we focus on the logistic case where the link function is defined as $g(E(y)) = \text{logit}(E(y)) = \log(E(y)/(1 - E(y)))$.

For each of the Boolean functions L_i, $1 \leq i \leq t$, in the above model, we define the *order* of the function as the minimum number of variables $x_1, \ldots, x_m$ that are sufficient to determine the value of L_i. For example, a function that depends on only one of the variables is said to be first order, and so on. The order of the model is defined as the maximum order of all the functions involved. As mentioned above, the basic logistic regression model is usually defined as a first order model that includes all the m first order functions $L_i(\boldsymbol{x}) = x_i$, $1 \leq i \leq t = m$.

1.1 Related Work

An important prior work regarding logical features was done by Ruczinski et al. in [11] who also provide an implementation of their method in the R package `LogicReg`.[1] It uses a greedy algorithm to search through the space of possible Boolean functions, with additional simulated annealing step to avoid local optima. It shows better performance than the tree based or rule based methods that are used by CART [2] and MARS [4]. However, as the number of covariates and the order of interactions are increased, the number of possible Boolean functions grows significantly. This makes greedy search heuristics computationally inefficient and prone to converge to local optima.

On the other hand, Shi et al. [13] proposed a Lasso based method which is suitable for identifying a large set of interactions. However, they only interactions defined using the AND operator. To allow more types of Boolean operations, the works in [1,8] incorporate extra information such as the structure of coefficients to guide the learning process. However, these methods need expert knowledge that is not always available. They are also unable to handle complex interactions. Both methods are only suitable for cases when all coefficients inside a group are either zero or non-zero. Our new Lasso based method can efficiently and

[1] Available from CRAN, http://cran.r-project.org.

effectively learn arbitrary logical functions and deal with situations when the covariates have multiple values.

The least absolute shrinkage and smoothing operator (Lasso) for linear regressions was proposed by Tibshirani in 1996 [15] and has since then gained a lot of popularity in subset selection problems. Lasso aims to minimize the sum of squared errors subject to a bound on the sum of absolute values of the regression coefficients, i.e., the L_1 norm of the coefficient vector $\boldsymbol{\beta} = (\beta_1, \ldots, \beta_t)^T$:

$$\operatorname*{argmin}_{\boldsymbol{\beta} \,:\, \|\boldsymbol{\beta}\|_1 < \lambda} \|Y - X\boldsymbol{\beta}\|_2^2, \tag{2}$$

where $Y = (y^{(1)}, \ldots, y^{(n)})^T$ is a vector of n responses and X is a matrix whose rows are n observation vectors $\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(n)}$, and $\lambda > 0$ is a regularization parameter.

Lasso encourages sparsity in the estimated coefficient vector. As the value of λ is decreased, more and more coefficients are set to zero. This feature is especially useful when we need to identify a small set of relevant variables out of a large collection of candidates. Thus Lasso is suitable for discovering interactions in regression problems. Furthermore, there are plenty of well-developed tools for solving Lasso problems efficiently.

If we encode logical features constructed using the original variables as a new set of regressor variables, we can use Lasso to select the significant ones out of all possible interactions. However, the number of all possible logical expressions grows too rapidly to be handled efficiently. Furthermore, the redundancy caused by different combinations of logical formulas expressing the exact same model may lead to numerical and statistical instability. To restrict the number of predictors within a manageable size, previous work in [13] confines the logic expressions to include only two variables and the AND operator.

1.2 Contributions

In this paper we introduce a Lasso-based method for learning sparse logistic regression models with logical features. Technically, the method is implemented as a *transformed Lasso* [10] which involves a transformation matrix that multiplies the original design matrix X. The transformed Lasso can deal with any type of logical interactions. Here we also extend it to handle multivalued covariates. In the following sections, we first propose a transformation of the original data involving a generalized Walsh-Hadamard matrix. The transformation increases the dimension of the data but enables the learning of arbitrarily complex logical dependencies. We demonstrate the power of the proposed method by comparing its performance to that of the greedy method in LogicReg with different settings of sample sizes and model complexities. Finally, we propose several possible future extensions.

2 Model Formulation

As is well known, any Boolean formula can be decomposed as a linear combination of XOR functions of the same or lower orders as the formula itself. For example, to represent AND or OR formulas over a subset of indices $I_0 \subseteq \{1, 2, \dots, l\}$, by using linear combinations of XOR formulas, we can write them respectively as

$$\text{AND}(x_{I_0}) = 2^{1-|I_0|} \sum_{l=1}^{|I_0|} (-1)^{l-1} \sum_{I' \subseteq I_0, |I'|=l} \text{XOR}(\{x_{I'}\}), \tag{3}$$

$$\text{OR}(x_{I_0}) = 2^{1-|I_0|} \sum_{l=1}^{|I_0|} \sum_{I' \subseteq I_0, |I'|=l} \text{XOR}(\{x_{I'}\}). \tag{4}$$

To map a binary covariate matrix to a design matrix that is composed of XOR functions of subsets of the covariates, it is convenient to use the discrete Walsh-Hadamard transform, see [10]. To construct the design matrix, we first expand the original covariate matrix into a larger matrix with columns corresponding to indicator variables for each possible covariate vector (e.g., in the case of two binary variables: 00, 10, 01, and 11), and then (pre-)multiply this matrix by the Walsh-Hadamard matrix. For a more complete explanation including a detailed example, see [10]. As we explain below, in practice the matrices need not be explicitly constructed.

The rows of a Walsh-Hadamard matrix of order 2^m correspond to all possible vectors composed of the m covariates. The columns of a such matrix are XOR functions on all subsets of the covariates given by the corresponding row. For example, the rows of a fourth order Walsh-Hadamard matrix correspond to all combinations of two binary elements x_1 and x_2, while the columns are XOR(0), XOR(x_1), XOR(x_2), and XOR(x_1, x_2). The fourth order Walsh-Hadamard matrix is then

$$W_4 = \begin{array}{c} \\ 00 \\ 10 \\ 01 \\ 11 \end{array} \!\!\begin{array}{c} \begin{array}{cccc} \text{XOR}(0) & \text{XOR}(x_1) & \text{XOR}(x_2) & \text{XOR}(x_1, x_2) \end{array} \\ \left(\begin{array}{cccc} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 \end{array} \right) \end{array} . \tag{5}$$

When the covariates takes three or more values, we can formulate the design matrix in the similar way as forming the Walsh-Hadamard matrix. Each column corresponds to a XOR function of a subset of possible sequences while each variable equals to one of its possible values. For instance, if the covariates are ternary with values $\{0, 1, 2\}$, the columns relating to two variables x_1 and x_2 are XOR(0), XOR($x_1 = 0$), sc xor($x_1 = 1$), XOR($x_2 = 0$), XOR($x_2 = 1$), XOR($x_1 = 0, x_2 = 0$), XOR($x_1 = 0, x_2 = 1$), XOR($x_1 = 1, x_2 = 0$), and XOR($x_1 = 1, x_2 = 1$). We ignore the XOR functions when x_1 or x_2 equals 2, because we can represent them as linear combinations of XOR functions when x_1 or x_2 equals 0 or 1. Such linear dependencies would significantly complicate the parameter estimation stage. The design matrix W'_9 based on the Walsh-Hadamard matrix for two ternary variables is

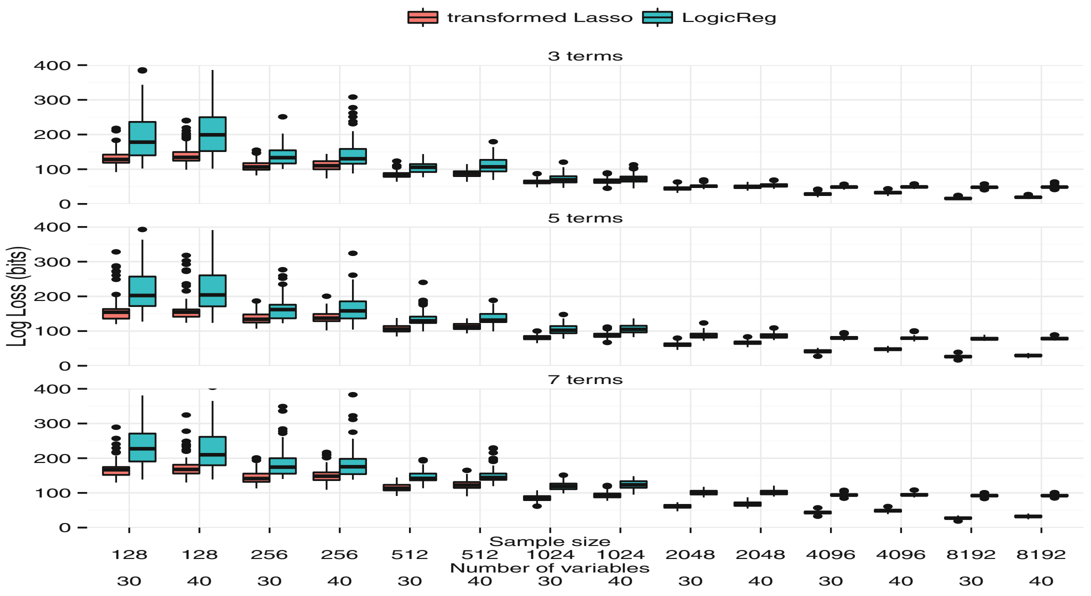
$$W'_9 = \begin{array}{c} 00 \\ 10 \\ 01 \\ 11 \\ 02 \\ 20 \\ 22 \\ 12 \\ 21 \end{array} \begin{pmatrix} 0 & 1 & 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \end{pmatrix}. \quad (6)$$

In practice, when the number of covariates is large, it becomes more likely that we only observe a small subset of all possible combinations of variables. In this case we do not need to build the full Walsh-Hadamard matrix. For each observed combination, we can map it to a vector of the corresponding XOR functions directly.

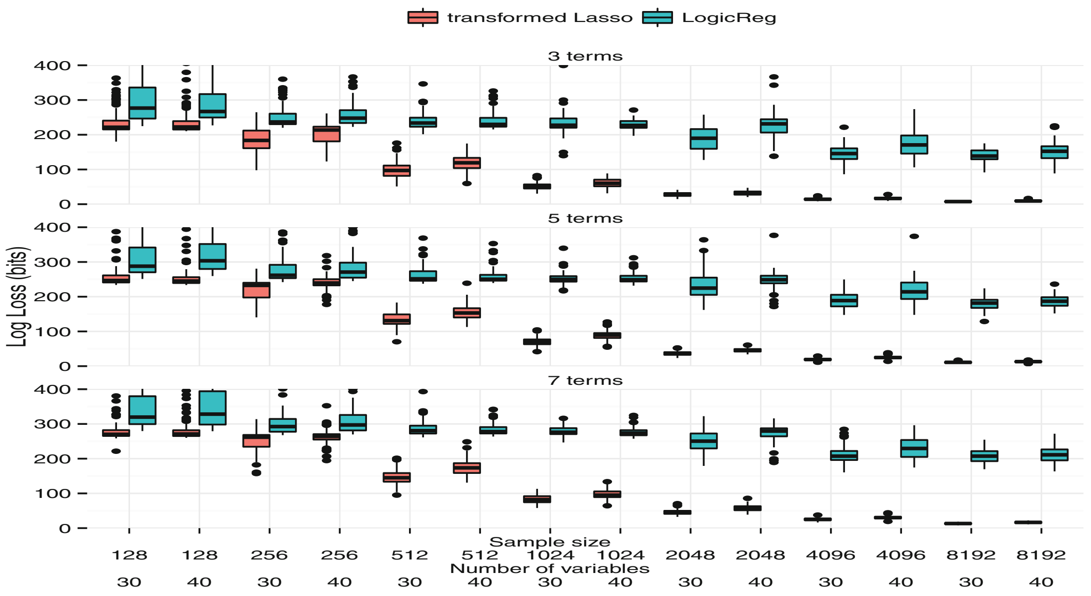
For example, assume that we have five binary variables $\{x_1, x_2, \ldots, x_5\}$, and the linear predictor only depends on the first three variables: $g(E(y)) = \beta_0 + \beta_1 x_1 + \beta_2 \text{AND}(x_2, x_3)$. If we restrict the maximum order of the interactions to be three, the design matrix has $\binom{5}{0} + \binom{5}{1} + \binom{5}{2} + \binom{5}{3} = 1 + 5 + 10 + 10 = 26$ columns corresponding to the XOR functions with at most third order interactions. If the sample size is sufficiently large, by using Lasso, only the coefficients for the five predictors: XOR(0), XOR(x_1), XOR(x_2), XOR(x_3), and XOR(x_2, x_3) will be non-zero. Then we can apply the XOR functions with non-zero coefficients in new data sets for prediction. In the following experiments on simulated data sets, we show that even when the number of XOR functions is relatively large, the learned models quickly converge toward the generating model as the sample size is increased.

3 Experiments

We compare the greedy search method in the R package `LogicReg` with transformed Lasso for different models and sizes of training data in three experiments. For Lasso regression, we use the popular R package `glmnet`. For each model and size of training data, we generated 100 different training data sets based on the true model. Later we compare the log-losses of learned models by LogicReg and transformed Lasso by evaluating them on another 100 new data sets with sample sizes 1024. We ran all the experiments on computers with 32 GB RAM and 2.53 GHz CPUs using only a single core at a time.

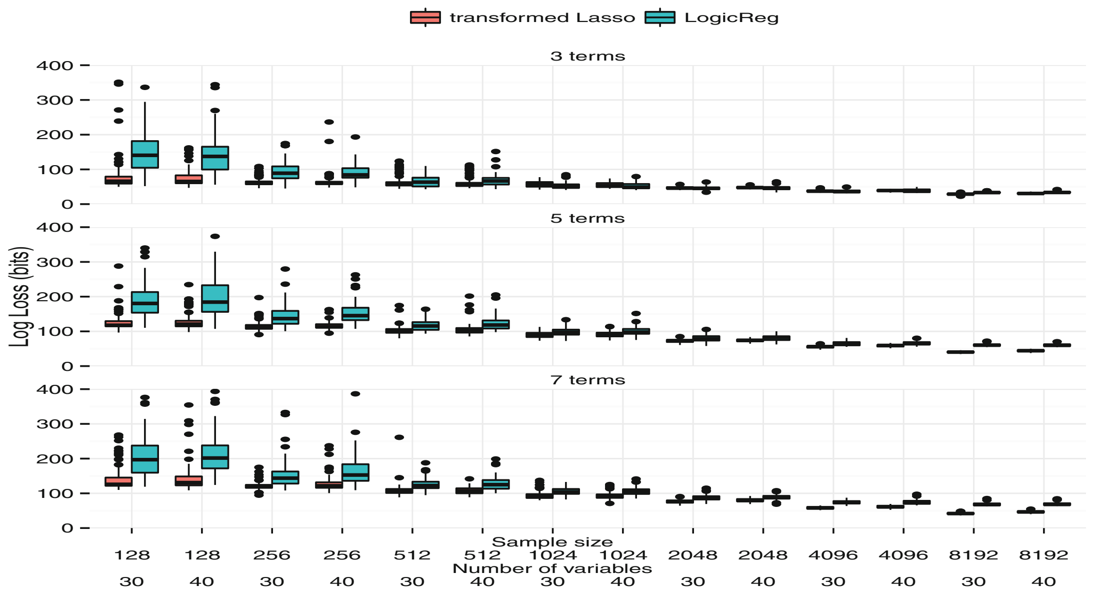


(a)



(b)

Fig. 1. (a) Box plot of log-losses for independent test data by LogicReg and transformed Lasso when data is simulated from Eq. 7(a) (3 terms)–(c) (7 terms), respectively. Sample sizes increase along the x-axis, and for each sample size, the total number of regressor variables is either 30 or 40 as indicated in the x-axis label. Upper and lower whiskers show the maximum and minimum values of log-losses respectively. To emphasize the differences between the log-losses of the two methods, the outliers (mainly the results by LogicReg) that lie above the upper limit are not shown. (b) Data simulated from Eq. 8(a)–(c) that include XOR operators. (c) Box plot of log-losses of inferred logic functions by LogicReg and transformed Lasso for models in Eq. 9(a)–(c).



(c)

Fig. 1. (*continued*)

3.1 Experiment 1

First, we use the true models:

$$\begin{aligned}
\text{logit}(E(Y)) = {} & 0.5 - 1.3\,\text{OR}[\text{AND}(\neg x_1, \neg x_2), x_3] \\
& + 1.5\,\text{OR}(\neg x_4, \neg x_5, \neg x_6) - 1.7\,\text{AND}(x_7, x_8, x_9), \quad (7a) \\
\text{logit}(E(Y)) = {} & 0.5 - 1.3\,\text{OR}[\text{AND}(\neg x_1, \neg x_2), x_3] \\
& + 1.5\,\text{OR}(\neg x_4, \neg x_5, \neg x_6) - 1.7\,\text{AND}(x_7, x_8, x_9) \\
& + 1.1\,\text{AND}(\neg x_{10}, \neg x_{11}, \neg x_{12}) - 0.9\,\text{OR}(x_{13}, x_{14}, x_{15}), \quad (7b) \\
\text{logit}(E(Y)) = {} & 0.5 - 1.3\,\text{OR}[\text{AND}(\neg x_1, \neg x_2), x_3] \\
& + 1.5\,\text{OR}(\neg x_4, \neg x_5, \neg x_6) - 1.7\,\text{AND}(x_7, x_8, x_9) \\
& + 1.1\,\text{AND}(\neg x_{10}, \neg x_{11}, \neg x_{12}) - 0.9\,\text{OR}(x_{13}, x_{14}, x_{15}) \\
& + 0.7\,\text{OR}[\text{AND}(x_{16}, x_{17}), x_{18}] - 0.5\,\text{AND}(x_{19}, x_{20}, x_{21}). \quad (7c)
\end{aligned}$$

The linear predictors contain three, five or seven separate terms (not including intercepts) that have no XOR operators. The terms included in the simpler linear predictors (three and five terms) are subsets of the terms in the more complex linear predictors. Each term in the linear predictor involves three covariates. The covariates are independent from each other and generated with equal probabilities for 1s and 0s.

For the LogicReg method, we restrict the search space by no more than three variables in a term and no more than five separate terms.[2] Model selection

[2] We found that the implementation of LogicReg provided in the package cannot handle more than five terms in the same formula.

is performed by 10-fold cross validation. For the method based on the Lasso framework, we also consider only up to third order interactions. The best tuning parameter λ in Lasso is also determined by 10-fold cross validation. The sample size ranges from 64 to 8192. The total number of covariates is 30 or 40 of which all but the ones appearing in Eq. 7(a)–(c) have no effect on the response. For example, the response of model in Eq. (7a) only depends on nine variables, $x_1, \ldots, x_9$, while covariates $x_{10}, \ldots, x_m$ ($m = 30$ or 40) are irrelevant. We repeat the experiment on 100 different training sets for each combination of sample size and number of covariates. The log-likelihoods achieved by both methods are compared to the log-likelihoods under the true model on 100 independently generated samples of size 1024.

We show how the log-loss changes under different conditions in Fig. 1a. We can see that when the sample size reaches 8192, both methods have almost converged to the same estimated models with small log-loss. The Lasso's ability to shrink most of the unimportant coefficients plays a key role in achieving a similar level of performance already from the small samples sizes, unlike the LogicReg method. For instance, with 40 covariates, selecting the required number of covariates (under 100) out of the 10 701 candidates seems to be very hard for LogicReg up until sample size 1024.

When the sample size grows, both methods have increasingly better results in the sense of both smaller average log-losses and smaller variance between the repetitions. But when the sample size is relatively small, the greedy search method is very unstable. This is because it needs to pick the right Boolean terms out of a much larger number of possible terms — recall that the number of features considered in the LogicReg method is significantly greater than in transformed Lasso because LogicReg includes all possible Boolean operators while the latter method only includes XOR operators. Moreover, the Lasso method has no local optima unlike the greedy search applied in LogicReg.

The performance of the two methods also depends on the number of irrelevant covariates. This is because the number of terms to be considered in both methods grows with the number of covariates. For example, the number of XOR terms included in the design matrix in transformed Lasso for 30 variables is 4526 and for 40 variables 10 701. On the other hand, the (negative) effect of increasing the number of covariates is not very significant compared to the (positive) effect of increasing the sample size except in the sense that as the number of variables is increased, the computational cost of both methods increases sharply.

3.2 Experiment 2

In the second experiment, we replace the AND and OR operators in Eq. (7) by the XOR operator while keeping the coefficients unchanged. The new data generating models are

$$\begin{aligned}\text{logit}(E(Y)) = 0.5 &- 1.3\,\textsc{xor}(\neg x_1, \neg x_2, x_3) \\ &+ 1.5\,\textsc{xor}(\neg x_4, \neg x_5, \neg x_6) - 1.7\,\textsc{xor}(x_7, x_8, x_9),\end{aligned} \tag{8a}$$

$$\begin{aligned}\text{logit}(E(Y)) = 0.5 &- 1.3\,\text{XOR}(\neg x_1, \neg x_2, x_3)\\ &+ 1.5\,\text{XOR}(\neg x_4, \neg x_5, \neg x_6) - 1.7\,\text{XOR}(x_7, x_8, x_9)\\ &+ 1.1\,\text{XOR}(\neg x_{10}, \neg x_{11}, \neg x_{12}) - 0.9\,\text{XOR}(x_{13}, x_{14}, x_{15}), \qquad (8b)\end{aligned}$$

$$\begin{aligned}\text{logit}(E(Y)) = 0.5 &- 1.3\,\text{XOR}(\neg x_1, \neg x_2, x_3)\\ &+ 1.5\,\text{XOR}(\neg x_4, \neg x_5, \neg x_6) - 1.7\,\text{XOR}(x_7, x_8, x_9)\\ &+ 1.1\,\text{XOR}(\neg x_{10}, \neg x_{11}, \neg x_{12}) - 0.9\,\text{XOR}(x_{13}, x_{14}, x_{15})\\ &+ 0.7\,\text{XOR}(x_{16}, x_{17}, x_{18}) - 0.5\,\text{XOR}(x_{19}, x_{20}, x_{21}). \qquad (8c)\end{aligned}$$

All other experiment settings are the same as in the first experiment. Comparing the results of the second experiment as illustrated in Fig. 1b with the first experiment, we find that for small sample sizes, the transformed Lasso method has difficulty in fitting the true models if they contain many XOR operators. However, it performs better for sample sizes larger than 1024. On the other hand, the greedy search method fails to find acceptable models in the second experiment even when the sample size is 8196.

For the models in the second experiment, we need less non-zero coefficients in the transformed Lasso method, because it contains only XOR functions. For instance, if we have three input terms, we only need three predictors with non-zero coefficients to represent the whole linear predictor. But for the linear predictor of three terms in the first experiment, we need to decompose it into fifteen XOR functions. There are less correct XOR terms in the second experiment, thus each XOR term is comparatively more important. For small sample sizes, when there is not enough information, it becomes much harder to identify the correct XOR functions. For the linear predictors in the second experiment, any incorrect choice of XOR term decreases the accuracy much more significantly than in the first experiment. Therefore, transformed Lasso performs worse when the sample size is small in the second experiment than in the first experiment. On the other hand, a smaller set of non-zero coefficients result in stronger effects of the relevant coefficients on responses. When we have enough training data, it becomes easier for transformed Lasso to identify the right terms in the second experiment.

On the other hand, the greedy search method in LogicReg represents Boolean functions only by AND, OR or negation operators. Thus, it needs to construct much more complex expressions in the second experiment. For example, the term XOR(x_1, x_2, x_3) needs to be expressed by interactions between ten variables with the repeating use of x_1, x_2 and x_3. It requires exploring an extremely large model space. Moreover, the greedy search method starts by picking up the most significant variables. However, a single variable in a XOR function has a much weaker effect in responses, which makes it very hard for the greedy search method to find a good starting point. Furthermore, in the following process, it can only modify a current model by adjusting one variable or one operator at each step. Although the simulated annealing method are incorporated to avoid local optima, an updated model should be reachable by a single move from the previous one. Therefore, a proper starting point is crucial for the greedy search method, which is difficult to find in the second experiment. Even when the sample size is as large as 8192, the learned models by the greedy search method are far from close to the true ones.

3.3 Experiment 3

For the last experiment, we show how the two methods work when variables have multiple values. We use the similar data generating models as in the first experiment, but allow the variables to take one of the three values: $\{0, 1, 2\}$. The true models are:

$$\begin{aligned}\mathrm{logit}(E(Y)) &= 0.5 - 1.3\,\textsc{or}[\textsc{and}(x_1 \neq 0, x_2 \neq 1), x_3 = 2] \\ &\quad + 1.5\,\textsc{or}(x_4 \neq 0, x_5 \neq 1, x_6 \neq 2) - 1.7\,\textsc{and}(x_7 = 0, x_8 = 1, x_9 = 2), \qquad (9a)\\ \mathrm{logit}(E(Y)) &= 0.5 - 1.3\,\textsc{or}[\textsc{and}(x_1 \neq 0, x_2 \neq 1), x_3 = 2] \\ &\quad + 1.5\,\textsc{or}(x_4 \neq 0, x_5 \neq 1, x_6 \neq 2) - 1.7\,\textsc{and}(x_7 = 0, x_8 = 1, x_9 = 2) \\ &\quad + 1.1\,\textsc{and}(x_{10} \neq 0, x_{11} \neq 1, x_{12} \neq 2) - 0.9\,\textsc{or}(x_{13} = 0, x_{14} = 1, x_{15} = 2), \qquad (9b)\\ \mathrm{logit}(E(Y)) &= 0.5 - 1.3\,\textsc{or}[\textsc{and}(x_1 \neq 0, x_2 \neq 1), x_3 = 2] \\ &\quad + 1.5\,\textsc{or}(x_4 \neq 0, x_5 \neq 1, x_6 \neq 2) - 1.7\,\textsc{and}(x_7 = 0, x_8 = 1, x_9 = 2) \\ &\quad + 1.1\,\textsc{and}(x_{10} \neq 0, x_{11} \neq 1, x_{12} \neq 2) - 0.9\,\textsc{or}(x_{13} = 0, x_{14} = 1, x_{15} = 2) \\ &\quad + 0.7\,\textsc{or}[\textsc{and}(x_{16} = 0, x_{17} = 1), x_{18} = 2] - 0.5\,\textsc{and}(x_{19} = 0, x_{20} = 1, x_{21} = 2). \qquad (9c)\end{aligned}$$

To build the design matrix for the transformed Lasso method, we code interactions between ternary covariates as described in Sect. 2. On the other hand, because the method used by LogicReg requires binary input, we add dummy variables to indicate when a covariate takes one of the three values.

Figure 1c shows that for ternary covariates, transformed Lasso works better than the greedy search method under all sample sizes and numbers of covariates. The ternary case is more difficult for both methods than the binary case because it has much larger search spaces for both methods. However, both methods still show their power to learn good model if there are enough data. When the sample size reaches 4096, both methods converge to true models for ternary covariates as well as for binary covariates.

Based on the previous experiments, we can see that transformed Lasso performs better than the greedy search method in all the cases with models of different complexities and training sample sizes. The greedy search method achieves reasonable results only when there is a decent number of samples and input functions are simple. However, it has very large log-losses as well as large variances when the sample size is less than 512 in all different settings in the three experiments. But in real life studies, a relative small number of training samples is very common. Moreover, the greedy search method fails when the interaction includes complex operators like XOR, which makes the responses less affected by any single covariate involved in the interaction.

Another factor that we need to consider is computational cost. For the simple case with three terms in the linear predictor, a total of 30 covariates and sample size 128 in the first experiment, we need on average 373 s to perform model learning by the greedy search method in LogicReg, but only 6.0 s for the transformed Lasso method. For the more complex model with seven terms including XOR operators, a total of 40 covariates and sample size 8196 in the second experiment, LogicReg uses on average 15 920 s, whereas the Lasso based method needs only 1 580 s.

4 Discussion

In this study, we propose on approach to learning sparse logistic models with logical features of multivalued inputs. The same approach can also be applied in other types of regression models such as Poisson regression and Cox proportional hazards models. In our experiments with simulated data, our Lasso based method was able to estimate different models based on AND, OR, and XOR features and their combinations more effectively than the earlier LogicReg method. More extensive experiments, including a comparison to other types of state-of-the-art classification techniques will provide more detailed information about the performance of the proposed method.

In future work, the proposed approach can be extended in several directions. Firstly, for handling a large number of variables, we can use the `LIBLINEAR` [3] library that scales better for large sparse data. Even then, a very large number of covariates will necessarily pose problems to methods that include high order interactions. For example, most genome wide studies may include hundreds of thousands genomic covariates. Many existing approaches include a screening stage to narrow down the set of candidate covariates to a manageable number. This can be done either by exploiting expert knowledge or by other statistical methods, see, e.g. [6,12]. Exploring suitable screening methods for the transformed Lasso with very high dimensional data is an interesting research direction that is necessary for many genomics applications.

Moreover, it can be helpful to integrate additional assumptions concerning the model structure to guide the learning process in the spirit of [1,8]. This can be achieved by modifying the Lasso penalization in various ways. For instance, it may be reasonable to assume that if a given high order coefficient takes a non-zero value, the lower order interactions among the variables that are included in the higher order interactions are more likely to be non-zero as well. Different group Lasso techniques are available for achieving this [9]. Furthermore, Lasso tends to select most significant variables out of a group of correlated variables. Integrating the structure of predictors can also be beneficial in the case when the covariates are highly correlated.

Acknowledgments. We thank Mr. Jussi Määttä for checking optimization results and the anonymous reviewers for useful comments. This work was supported in part by the Academy of Finland (Centre-of-Excellence COIN) and by the DoCS graduate school of the Department of Computer Science at the University of Helsinki.

References

1. Bien, J., Taylor, J., Tibshirani, R.: A lasso for hierarchical interactions. Ann. Appl. Stat. **41**(3), 1111–1141 (2013)
2. Breiman, L.: Bagging predictors. Mach. Learn. **24**(2), 123–140 (1996)
3. Fan, R.E., Chang, K.W., Hsieh, C.J., Wang, X.R., Lin, C.J.: LIBLINEAR: A library for large linear classification. J Mach. Learn. Res. **9**, 1871–1874 (2008)
4. Friedman, J.H.: Multivariate adaptive regression splines. Ann. Stat. **19**(1), 1–67 (1991)
5. Hastie, T., Tibshirani, R., Friedman, J.: The Elements of Statistical Learning: Data Mining, Inference and Prediction. Springer, New York (2009)
6. Holger, S., Ickstadt, K.: Identification of SNP interactions using logic regression. Biostat. **9**(1), 187–198 (2008)
7. Keleş, S., van der Laan, M.J., Vulpe, C.: Regulatory motif finding by logic regression. Bioinform. **20**(16), 2799–2811 (2004)
8. Kim, S., Xing, E.P.: Tree-guided group lasso for multi-response regression with structured sparsity, with an application to eQTL mapping. Ann. Appl. Stat. **6**(3), 1095–1117 (2012)
9. Meier, L., van de Geer, S., Bühlmann, P.: The group lasso for logistic regression. J. Roy. Stat. Soc B. **70**(1), 53–71 (2008)
10. Roos, T., Yu, B.: Estimating sparse models from multivariate discrete data via transformed Lasso. In: Proceedings of Information Theory and Applications Workshop, pp. 290–294. IEEE Press (2009)
11. Ruczinski, I., Kooperberg, C., LeBlanc, M.: Logic regression. J. Comp. Graph Stat. **12**(3), 475–511 (2003)
12. Saeys, Y., Inza, I., Larrañaga, P.: A review of feature selection techniques in bioinformatics. Bioinform. **23**(19), 2507–2517 (2007)
13. Shi, W., Wahba, G., Wright, S., Lee, K., Klein, R., Klein, B.: LASSO-Patternsearch algorithm with application to ophthalmology and genomic data. Stat Interface. **1**(1), 137–153 (2008)
14. Suehiro, Y., Wong, C.W., Chirieac, L.R., Kondo, Y., Shen, L., Webb, C.R., et al.: Epigenetic-genetic interactions in the APC/WNT, RAS/RAF, and P53 pathways in colorectal carcinoma. Clin. Cancer Res. **14**(9), 2560–2569 (2008)
15. Tibshirani, R.: Regression shrinkage and selection via the lasso. J. Roy. Stat. Soc B. **58**(1), 267–288 (1996)
16. Zhao, J., Li, J., Xiong, M.: Test for interaction between two unlinked loci. Am. J. Hum. Gen. **79**(5), 831–845 (2006)