Using Probabilistic Knockoffs of Binary Variables to Control the False Discovery Rate

Aaron Maurer

July 12, 2015

1 Introduction

Variable selection is an essential problem to fitting a regression model. For linear regression, the knockoff filter (Barber and Candès, 2014) offers a method of exact FDR control. However, the method as originally formulated does not extend to other generalized linear models such as logistic regression. This paper offers an extension of this method which will work with binary features in the context of linear regression, but also will extend to other GLMs.

1.1 Knockoff Filter

Let us consider the consider the usual setting for linear regression, where n observations of a variable of interest y arise from the model

$$\mathbf{y} = X\beta + \mathbf{z}$$

where $\mathbf{y} \in \mathbb{R}^n$ are the observed values of y, $X \in \mathbb{R}^{n\times p}$ is the matrix of predictor variables, $\beta \in \mathbb{R}^p$ are the unknown coefficients, and $\mathbf{z}$ is Gaussian noise. For the purpose of this paper, only the case where $2p \leq n$ will be considered, though the methods can be extended to $p \leq n$. I will to refer the jth column of X as X_j, which is n observations of the random variable x_j. The random variable x_j is in turn jth entry of the random vector valued variable $\mathbf{x}$. In the situation where β is sparse and there is reason to expect that x_j and y are uncorrelated for many j, the knockoff filter is a method to select a subset of the x_j which are likely correlated with y.

The method specifically seeks to control the false discovery rate (FDR), which is the expected portion of some selection of x_j which are, in fact, uncorrelated with y. For a procedure which chooses a subset of variables $\hat{S} \subseteq \{1, \ldots, p\}$, this is defined as

$$\text{FDR} = \text{E}\left[\frac{|\{j : \beta_j = 0 \;\&\; j \in \hat{S}\}|}{\max\{|\hat{S}|, 1\}}\right]$$

FDR is controlled at level q if FDR is less than q irrespective of the coefficients β.

The knockoff filter achieves FDR control in several steps, the first of which is creating a set of 'knockoff' features $\tilde{X}_j$ which imitate the original X_j while being less correlated with y. In particular, the matrix of knockoffs

$\tilde{X}$ has the same internal correlation structure as X, the knockoff feature $\tilde{X}_j$ has the same correlation with other X_i as X_j does, but the correlation between $\tilde{X}_j$ and X_j is minimized. In other words, where $\hat{\Sigma} := X^T X$,

$$\tilde{X}^T \tilde{X} = \hat{\Sigma} \text{ and } \tilde{X}^T X = \Sigma - \text{diag}\{\mathbf{s}\}$$

for some vector $s \in \mathbb{R}^p$ such that, where $\text{diag}\{\Sigma\}$ is the vector made from the main diagonal of Σ, $\text{diag}\{\Sigma\} - \mathbf{s}$ is small. Since $\tilde{X}_j$ and X_j have relatively low correlation, as long as X_j is created independently of $\mathbf{y}$, $\tilde{X}_j$ will have lower correlation with $\mathbf{y}$ than X_j. In the case where X_j is uncorrelated with $\mathbf{y}$, $\tilde{X}_j$ will also be uncorrelated with $\mathbf{y}$. How these knockoffs are actually generated will be described shortly.

The second step is to fit a series of Lasso model of $\mathbf{y}$ on the combined design matrix $[X\ \tilde{X}]$ so as to determine the largest value λ at which the coefficient for each X_j and $\tilde{X}_j$ is nonzero.[1] Recall, for a given λ, the estimated coefficient from the Lasso regression will be

$$\beta(\lambda) = \text{argmin}_{\mathbf{b}} \left\{ \frac{1}{2} \|\mathbf{y} - [X\ \tilde{X}]\mathbf{b}\|_2^2 + \lambda \|b\|_1 \right\}$$

In the general case, the coefficient for one feature in a Lasso regression being first nonzero for a larger value of λ than another feature is an indication that the first feature is a stronger predictor of the outcome than the second. Thus, since, by construction, the knockoff features are weaker predictors than the originals, we would expect the original features to have nonzero coefficients sooner than the knockoff features when the original is a valid predictor. On the other hand, since $\beta(\lambda)$ only depends on $[X\ \tilde{X}]$ through the sufficient statistics $[X\ \tilde{X}]^T[X\ \tilde{X}]$ and $[X\ \tilde{X}]^T\mathbf{y}$, for null predictors, the coefficients of the original features won't enter any sooner on average than knockoff feature. This can be seen by switching a null X_j with $\tilde{X}_j$; by construction, $[X\ \tilde{X}]^T[X\ \tilde{X}]$ will be unaltered, while $\text{E}\left[[X\ \tilde{X}]^T\mathbf{y}\right]$ will also be unchanged, since $\text{E}[X_j^T\mathbf{y}] = \text{E}[\tilde{X}_j^T\mathbf{y}] = 0$.

This observation leads to the final step. Let Z_j be the largest λ such that feature j has a nonzero coefficient and $\tilde{Z}_j$ the largest λ such that knockoff j has a nonzero coefficient. Then, define W_j as

$$W_j = \begin{cases} Z_j & \text{if } Z_j > \tilde{Z}_j \\ -\tilde{Z}_j & \text{if } Z_j < \tilde{Z}_j \\ 0 & \text{if } Z_j = \tilde{Z}_j \end{cases}$$

When X_j is a null predictor, W_j will be symmetrically distributed around zero, and when X_j is true predictor, it will have a distribution skewed positive. Since W_j with large positive values are more likely to be true predictors, the knockoff filter selects the variables $\{j : W_j \geq T\}$, where T is a threshold defined as

$$T = \min \left\{ t > 0 \ : \ \frac{|\{j : W_j \leq -t\}|}{\max\{|\{j : W_j \geq t\}|, 1\}} \leq q \right\}$$

Basic logic of this method is fairly simple. Let $N_h(t)$ be the random variable for the number of null predictors with $W_j \leq t$, $N_l(t)$ be the number of null predictors with $W_j \geq t$, and $V_l(t)$ & $V_h(t)$ be the similar number of true

[1]Other statistics besides the λs can also be used in the knockoff method, but are not considered in this paper

predictors. Since the null W_j are symmetrically distributed around 0, $\mathrm{E}[N_l(t)] = \mathrm{E}[N_h(t)]$. Thus,

$$\begin{aligned}
q &\geq \mathrm{E}\left[\frac{|\{j : W_j \leq -t\}|}{\max\{|\{j : W_j \geq t\}|, 1\}}\right] \\
q &\geq \mathrm{E}\left[\frac{N_l(t) + V_l(t)}{\max\{N_h(t) + V_h(t), 1\}}\right] \\
q &\geq \mathrm{E}\left[\frac{N_l(t)}{\max\{N_h(t) + V_h(t), 1\}}\right] \\
q &\geq \mathrm{E}\left[\frac{N_h(t)}{\max\{N_h(t) + V_h(t), 1\}}\right] \\
q &\geq \mathrm{FDR}
\end{aligned}$$

So FDR is controlled, and since $V_l(t)$ will tend to be small, it should be controlled fairly tightly.

1.2 Original Knockoff Features

The original formulation of the method offers two similar methods of constructing knockoffs. Both of these will, by construction, have exactly the property that

$$\tilde{X}^T\tilde{X} = \hat{\Sigma} \text{ and } \tilde{X}^T X = \Sigma - \mathrm{diag}\{\mathbf{s}\}.$$

For both methods, the first step is to normalize the matrix X such that $X_j^T X_j = 1$ for all j. Then, let the Gram matrix of $[X\ \tilde{X}]$ be

$$G := [X\ \tilde{X}]^T[X\ \tilde{X}] = \begin{bmatrix} \Sigma & \Sigma - \mathrm{diag}\{\mathbf{s}\} \\ \Sigma - \mathrm{diag}\{\mathbf{s}\} & \Sigma \end{bmatrix}$$

A, the Schur complement of Σ in G can be calculated as

$$A = 2\,\mathrm{diag}\{\mathbf{s}\} - \mathrm{diag}\{\mathbf{s}\}\Sigma^{-1}\mathrm{diag}\{\mathbf{s}\}$$

For G to exist, G must be positive semi-definite, which happens if and only if A is positive semi-definite, which happens in turn if and only if[2]

$$\mathrm{diag}\{\mathbf{s}\} \succeq 0 \ \text{ and } \ 2\Sigma - \mathrm{diag}\{\mathbf{s}\} \succeq 0$$

Given this is true, A can be factored as $A = C^T C$. Combining this with a satisfactory $\mathbf{s}$ and an orthonormal matrix $\tilde{U} \in \mathbb{R}^{n \times p}$ such that $\tilde{U}^T X = 0$, a $\tilde{X}$ fulfilling the desired properties can be calculated as

$$\tilde{X} = X(I - \Sigma^{-1}\,\mathrm{diag}\{\mathbf{s}\}) + \tilde{U}C$$

As mentioned above, $\mathbf{s}$ should be chosen so as to make $\mathrm{diag}\{\Sigma\} - \mathbf{s}$ small. Since each X_j has been normalized, this means that $\mathrm{diag}\{\Sigma\} = \mathbf{1}$, so $\mathbf{1} - \mathbf{s}$ should be minimized in accordance with the restrictions on $\mathbf{s}$. The two methods differ in how $\mathbf{s}$ is chosen:

- *Equi-correlated knockoffs*: Each original features is set to have the same correlation with its knockoff by setting $\mathbf{s} = 2\min\{\lambda_{min}(\Sigma), 1\}\mathbf{1}$. For all simulations in this paper, this was the method used.

[2]A fleshed out version of this argument can be found in the original knockoff paper

- *SDP knockoffs*: The $\mathbf{s}$ which minimizes the average correlation between knockoff and original features can be found via a semi-definite programing problem:

$$\begin{aligned} \text{minimize} \quad & \|\mathbf{1} - \mathbf{s}\|_1 \\ \text{subject to} \quad & 0 \preceq \text{diag}\{\mathbf{s}\} \preceq 2\Sigma \end{aligned}$$

  This method is significantly more computationally intensive.

1.3 Binary Knockoffs

Though the "original" knockoff method, described above, achieves the exact desired knockoff properties, the individual values in the vector X_j will have little relation to the individual values in X_j. In particular, these values will often have very different empirical distributions. This effect is particularly noticeable when the random variable x_j is discrete but $\tilde{X}_j$ does not even consist of the same set of values. The end result is that for variable selection for other generalized linear models, which do not have the same sufficient statistics as linear regression which knockoffs are designed to hit, the original knockoffs will fail to control FDR.

Thus, this paper offers a new method of generating knockoffs which should offer superior performance with other GLMs. In this new method, the matrix of knockoff features $\tilde{X}$ will be generated by row from a random vector valued variable $\tilde{\mathbf{x}}$, each entry $\tilde{x}_j$ of which will have the property $\tilde{x}_j \sim x_j$. This random variable will also abide by a relaxation of the original knockoff condition:

$$\text{E}[\tilde{\mathbf{x}}^T\tilde{\mathbf{x}}] = \hat{\Sigma} \;\text{ and }\; \text{E}[\tilde{\mathbf{x}}^T\mathbf{x} \,\big|\, \mathbf{x}] = \Sigma - \text{diag}\{\mathbf{s}\}$$

This will ensure that, at the very least, these knockoffs are a suitable replacement for the original knockoffs for linear regression.

Generating such knockoffs for a completely general case of any $\mathbf{x}$ is likely infeasible. However, in the specific case where $\mathbf{x}$ is a random binary vector variable, this paper will demonstrate a method to generate such knockoff variables. Since binary data is extremely common in many areas, this will hopefully serve as a useful tool.

1.4 Paper Organization

The rest of this paper will be organized in the following fashion:

- Section 2 will go into more detail about how the original knockoffs breakdown with generalized linear models. In particular, several simulations will demonstrate how they breakdown.

- Section 3 will develop the method and theory for generating binary knockoffs.

- Section 4 will discuss tests of Binary knockoffs in simulation, both in comparison to the original knockoffs with Lasso and for logistic regression by itself.

- Section 5 is the final section, and will contain discussion of the results as well as areas for further work.

2 Issues With Deterministic Knockoffs

Note: I will try to hold to the convention that X is the $n \times p$ data matrix, while $\mathbf{x}$ is the random vector variable from which each row of X was drawn. Accordingly, $\tilde{X}$ will be the knockoff matrix while $\tilde{\mathbf{x}}$ is a random vector variable, at least when $\tilde{X}$ is generated randomly. x_i will be the random variable corresponding to the ith entry of $\mathbf{x}$, while X_i is ith column of X, with observations drawn from x_i. $\hat{\Sigma} = \frac{1}{n} X^T X$ is the empirical covariance matrix associated with X, while $\Sigma = \mathrm{E}(\mathbf{x}\mathbf{x}^T)$ is the theoretical covariance matrix associate with $\mathbf{x}$. Also, for a square matrix A, $\mathrm{diag}\{A\}$ is the vector of values along the diagonal, while for a vector $\mathbf{a}$, $\mathrm{diag}\{\}a$ is a square diagonal matrix with $\mathbf{a}$ along the diagonal.

Simulations with deterministic knockoffs reveal that they don't perform well in L1 regularized logistic regression. Even when X_i is a null predictor of y, the X_i still tend to enter the model prior to $\tilde{X}_i$. The issue is that even when $x \sim N_p(\mathbf{0}, \Sigma)$ for some $\Sigma \succeq 0$, the $\tilde{X}_i$ do not fit a normal distribution. This can be seen below in figure 1, where normal Q-Q plots of simulated variables X_i and corresponding knockoffs $\tilde{X}_i$ are provided. Of course, when X is a binary vector, $\tilde{X}$ completely doesn't match its distribution. These

3 Random Bernoulli Knockoffs

My solution to the issues with the

generate $\tilde{\mathbf{x}}$ randomly such that it has the desired knockoff properties in expectation. In particular, both should have similar marginal densities, expectations, and second moments. However, $\tilde{\mathbf{x}} \mid \mathbf{x}$ should also have desired knockoff property that $\mathrm{E}(\tilde{\mathbf{x}}^T \mathbf{x} \mid \mathbf{x}) = \Sigma - \mathbf{s}$, where $\|\mathrm{diag}\{\Sigma\} - s\|$ is small. In the general case, this is likely infeasible; however, if $\mathbf{x}$ is a binary vector, as is often the case, we know we are dealing with a much more limited class of random variables, and it should be possible to randomly generate $\tilde{\mathbf{x}} \mid \mathbf{x}$ so as to have the desired properties. At worst, this method will provide a suitable replacement for deterministic $\tilde{\mathbf{x}}$ for use with LASSO, and if we are lucky, it will work reasonably for other regularized GLMs.

3.1 Random Bernoulli Generation

Thankfully, there has been a reasonable amount of work on how one can generate random Bernoulli vectors with some kind of correlation among among the values. A random Bernoulli vector $\mathbf{x}$ can be summarized by its first two moments: a mean vector $\mathrm{E}(\mathbf{x}) = \mathbf{m} \in (\mathbf{0}, \mathbf{1})^{\mathbf{p}}$ and cross-moment matrix $\mathrm{E}(\mathbf{x}\mathbf{x}^{\mathbf{T}}) = \mathbf{M} \in (\mathbf{0}, \mathbf{1})^{\mathbf{p} \times \mathbf{p}}$. Obviously, $m_i = \mathrm{P}(x_i = 1)$, $M_{ij} = \mathrm{P}(x_i = x_j = 1)$, and $\mathbf{m} = \mathrm{diag}\{M\}$. For an arbitrary symmetric M to be valid cross-moment matrix, $M - mm^T \succeq 0$, and

$$\max\{0, m_i + m_j - 1\} \leq M_{ij} \leq \min\{m_i, m_j\}$$

for all $i \neq j$[3]. Given a qualifying M, or observed X, there are a few ways of generating more random $\mathbf{x}$.

[3] "On parametric families for sampling binary data with specified mean and correlation" - http://arxiv.org/abs/1111.0576

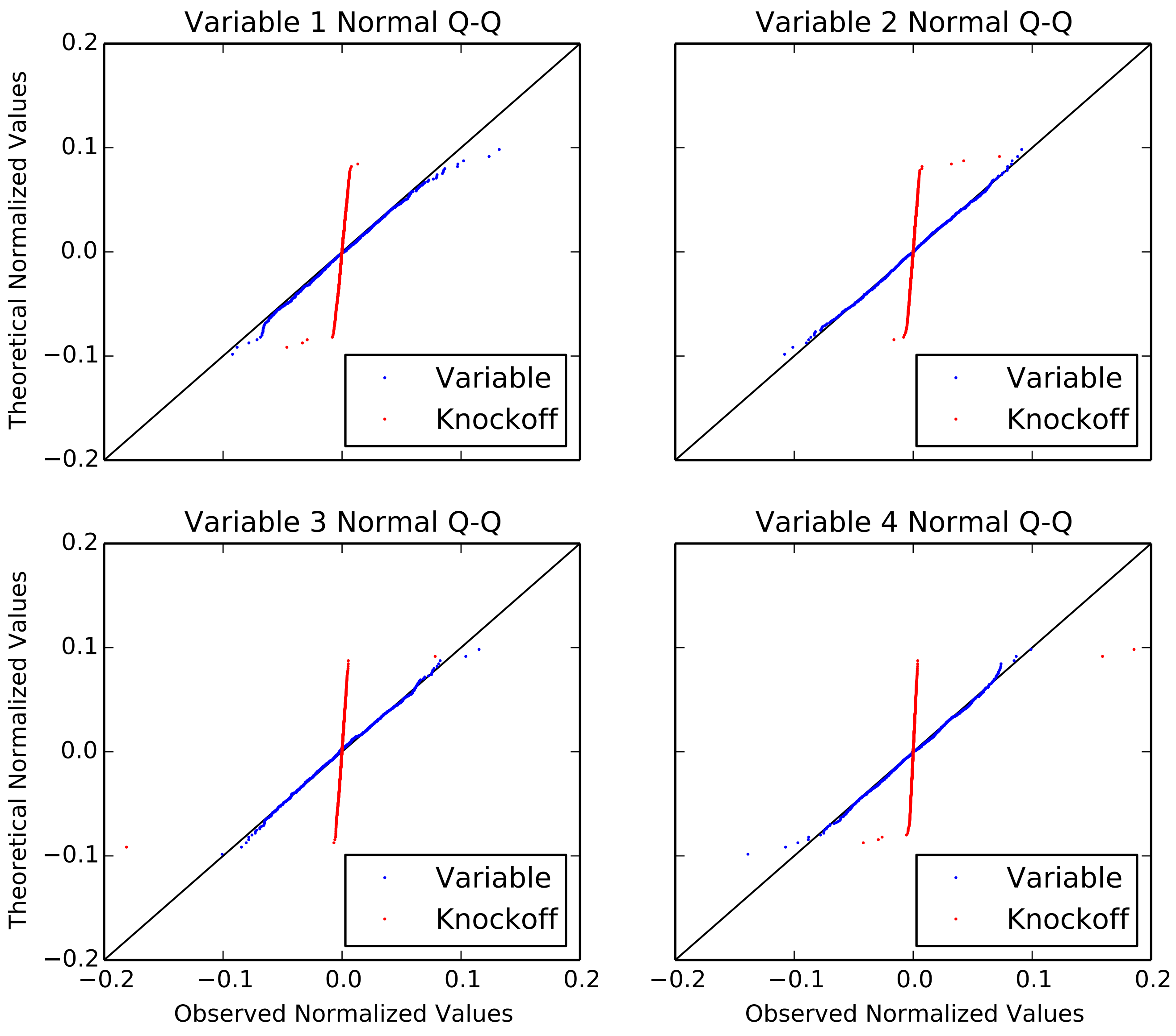

Figure 1: Normal Q-Q of original variables and knockoffs for simulation with 4 variables and 1,000 observations

3.1.1 Gaussian Copula Family

Since multivariate normal distributions are easy to randomly draw, the idea is to find some random normal variable $\mathbf{z} \sim N_p(\mathbf{0}, \Sigma)$ such that,for $x_i = I(z_i < 0)$, x has the desired properties. There are a number of ways to do this[4][5], but it turns out that there is only certain to exist a working Σ in the bivariate case.

[4] "On the Generation of Correlated Artificial Binary Data" - http://epub.wu.ac.at/286/1/document.pdf

[5] "On parametric families for sampling binary data with specified mean and correlation"

3.1.2 μ-Conditionals family

There exists a more flexible family which will always work for arbitrary M called μ-conditionals. The basic idea is that the X is generate sequentially as

$$\mathrm{P}(x_i = 1 \mid x_1, ..., x_{i-1}) = \mu\left(a_{ii} + \sum_{k=1}^{i-1} a_{ik} x_i\right)$$

for some monotone function $\mu : \mathbb{R} \to (0,1)$. This is essentially a binomial family GLM for a link function μ. If one takes all of the a_{kj}, they can form a lower triangular matrix A, and then the joint density can be expressed as

$$\mathrm{P}(\mathbf{x} = \gamma) \propto \mu(\gamma^T A \gamma)$$

If μ is chosen such that it is a bijection and differentiable, there is a unique A such that $\mathrm{E}(\mathbf{x}\mathbf{x}^T) = M$ when generated from this model[6]. The natural choice for μ is the logistic link function, which yields the Ising model, the "binary analogue of the multivariate normal distribution which is the maximum entropy distribution on $\mathbb{R}^p$ having a given covariance matrix." Additionally, it has the usual benefit that the coefficients can be viewed as a log odds ratio:

$$a_{ij} = \log\left(\frac{\mathrm{P}(x_j = x_k = 1)\mathrm{P}(x_j = x_k = 0)}{\mathrm{P}(x_j = 0, x_k = 1)\mathrm{P}(x_j = 1, x_k = 0)}\right)$$

when $i \neq j$. I think this dictates that if $\mathbf{x}$ is generated from this model with $a_{ij} = 0$, then x_i and x_j are independent.

There is no closed form to calculate the entries in A if $p > 1$, but they can be derived numerically two ways.

1. If one is attempting to replicate the empirical cross-moments from a data matrix X, a_{1i} to a_{ii} can be derived from fitting successive logistic regressions of X_i on $X_1 \dots X_{i-1}$ using maximum likelihood. a_{ji} for $i \neq j$ will then be the coefficient on X_j while a_{ii} is the intercept of the regression.

2. If one is just working with a desired cross-moment matrix M, the successive rows of A can be fit via Newton-Raphson.

   Let us say that the first $i-1$ rows have already been fit, resulting in the upper left $(i-1) \times (i-1)$ sub matrix A_{-i} of A. Let us say that $\mathbf{a}_i$ is the first i entries of the ith row of A (the rest will be 0 anyway). As well, let $\mathbf{m}_i$ be similarly the first i entries of the ith row of M. In other words, $\mathbf{m}_i = [\mathrm{E}(x_i x_j)]_{j=1}^{i}$. Finally,

[6] "On parametric families for sampling binary data with specified mean and correlation"

let us say that $\mathbf{x}_{-i}$ is the first $i-1$ entries of $\mathbf{x}$. We want to solve for $\mathbf{a}_i$ such that

$$\mathbf{m}_i = \mathrm{E}\left(x_i \begin{bmatrix} \mathbf{x}_{-i} \\ x_i \end{bmatrix}\right)$$

$$\mathbf{m}_i = \mathrm{E}\left(\mathrm{E}\left(x_i \begin{bmatrix} \mathbf{x}_{-i} \\ x_i \end{bmatrix} \Bigg| \mathbf{x}_{-i}\right)\right)$$

$$\mathbf{m}_i = \sum_{\mathbf{x}_{-i} \in \{0,1\}^{i-1}} \mathrm{P}(\mathbf{x}_{-i})\mathrm{P}(x_i = 1 \mid \mathbf{x}_{-i}) \begin{bmatrix} \mathbf{x}_{-i} \\ 1 \end{bmatrix}$$

$$\mathbf{m}_i = \sum_{\mathbf{x}_{-i} \in \{0,1\}^{i-1}} \frac{1}{c}\mu\left(\mathbf{x}_{-i}^T A_{-i}\mathbf{x}_{-i}\right)\mu\left(\mathbf{a}_i^T \begin{bmatrix} \mathbf{x}_{-i} \\ 1 \end{bmatrix}\right)\begin{bmatrix} \mathbf{x}_{-i} \\ 1 \end{bmatrix}$$

Where c is the appropriate normalizing constant. Let us define the quantity on the right in the last line as $f(\mathbf{a}_i)$. We can solve for $f(\mathbf{a}_i) = \mathbf{m}_i$ by successive Newton-Raphson iterations defined by

$$\mathbf{a}_i^{(k+1)} = \left[Hf\left(\mathbf{a}_i^{(k)}\right)\right]^{-1}\left[f\left(\mathbf{a}_i^{(k)}\right) - \mathbf{m}_i\right]$$

The Hessian matrix is calculated as

$$Hf\left(\mathbf{a}_i\right) = \sum_{\mathbf{x}_{-i} \in \{0,1\}^{i-1}} \frac{1}{c}\mu\left(\mathbf{x}_{-i}^T A_{-i}\mathbf{x}_{-i}\right)\mu'\left(\mathbf{a}_i^T \begin{bmatrix} \mathbf{x}_{-i} \\ 1 \end{bmatrix}\right)\begin{bmatrix} \mathbf{x}_{-i} \\ 1 \end{bmatrix}\begin{bmatrix} \mathbf{x}_{-i}^T & 1 \end{bmatrix}$$

With 2^{i-1} possible values for $\mathbf{x}_{-i}$, this can quickly become computationally expensive. Instead, with a series of values $\mathbf{x}_{-i}^{(k)} \sim \mathbf{x}_{-i}$, we can approximate

$$f\left(\mathbf{a}_i\right) \approx \frac{1}{K}\sum_{k=1}^{K}\mu\left(\mathbf{a}_i^T \begin{bmatrix} \mathbf{x}_{-i}^{(k)} \\ 1 \end{bmatrix}\right)\begin{bmatrix} \mathbf{x}_{-i}^{(k)} \\ 1 \end{bmatrix}$$

and

$$Hf\left(\mathbf{a}_i\right) \approx \frac{1}{K}\sum_{k=1}^{K}\mu'\left(\mathbf{a}_i^T \begin{bmatrix} \mathbf{x}_{-i}^{(k)} \\ 1 \end{bmatrix}\right)\begin{bmatrix} \mathbf{x}_{-i}^{(k)} \\ 1 \end{bmatrix}\begin{bmatrix} [x_{-i}^{(k)}]^T & 1 \end{bmatrix}$$

Though in theory A should always exist, in practice numerical issues may compound to the point that the Newton-Raphson method won't converge. In this case, one can solve instead for $\mathbf{m}_i^*(\tau)$, where, for $\tau \in [0,1]$

$$\mathbf{m}_i^*(\tau) = (1-\tau)\mathbf{m}_i + \tau\begin{bmatrix} 0 & \ldots & 0 & M_{ii} \end{bmatrix}^T$$

When $\tau = 0$, this yields the original problem, while when $\tau = 1$, it is treating x_i as independent of $\mathbf{x}_{-i}$. The latter will always have the solution

$$\mathbf{a}_i = \begin{bmatrix} 0 & \ldots & 0 & \log\left(\frac{M_{ii}}{1-M_{ii}}\right) \end{bmatrix}^T$$

The hope is that for some τ close to 0, convergence can be achieved, only causing a slight distortion from the desired cross moments.

3.2 Generating Binary Knockoffs

My method for generating binary knockoffs broadly involves two steps:

1. I use either the equal correlation method or the SDP method described in the original knockoff paper to find $\mathbf{s}$ such that $\|\text{diag}\{\hat{\Sigma}\} - \mathbf{s}\|$ is small and

$$\Sigma_L = \begin{bmatrix} \hat{\Sigma} & \hat{\Sigma} - \text{diag}\{\mathbf{s}\} \\ \hat{\Sigma} - \text{diag}\{\mathbf{s}\} & \hat{\Sigma} \end{bmatrix} \succeq 0$$

   I use a subscript L for large to denote items associated with the joint distribution of $[\mathbf{x}\ \tilde{\mathbf{x}}]$. If $\mathbf{m}_L = \text{E}\left([\mathbf{x}\ \tilde{\mathbf{x}}]\right)^T = [\mathbf{m}\ \mathbf{m}]^T$, then the desired cross moment matrix of the joint distribution is

$$M_L = \Sigma_L + \mathbf{m}_L \mathbf{m}_L^T$$

   To ensure that this is a valid cross moment matrix for a binary random vector is that

$$\max\{0, \mathbf{m}_{L,i} + \mathbf{m}_{L,j} - 1\} \leq M_{L,ij} \leq \min\{\mathbf{m}_{L,i}, \mathbf{m}_{L,j}\}$$

   I've built a check into the code for this, but in practice it shouldn't be a worry. This condition is always satisfied in a neighborhood of $M_{L,ij} = \mathbf{m}_{L,j}\mathbf{m}_{L,i}$. Since the we are either keeping the value $M_{L,ij}$ from a valid cross moment matrix or minimizing $|M_{L,ij} - \mathbf{m}_{L,j}\mathbf{m}_{L,i}|$, it would be very surprising if this condition was violated.

2. I can fit the matrix A that will generate random binary variables similarly to the method described in section 3.2 which have cross moments X_L. This can be used to generate the $\tilde{x}_i$ sequentially as $\tilde{x}_i \mid \mathbf{x}, \tilde{x}_1, \ldots, \tilde{x}_{i-1}$ so as to create $\tilde{X} \mid X$.

3.3 More Detail on Fitting A

I could fit A based on M_L exactly as described in the second method of 3.2, however, this isn't exactly what I do. First off, with p being potentially large, the simulation method for estimating $f(\mathbf{a})$ and $Hf(\mathbf{a})$ was the obvious choice. This involves fitting $\mathbf{a}_i$ based on the conditional distribution of $\mathbf{x}_{L,i}$ given randomly drawn partial vectors $\mathbf{x}_{L,-i}$. There is no need to simulate the marginal distribution of $\mathbf{x}$ though, since the simulation is only approximate and we already have a number of realizations in X. Thus, I only fit the lower half of A, using this process:

1. To get a simulation of size at least K, I create a matrix X_F (F for fixed) which is initially X stacked up until it has $K' \geq K$ rows.

2. For each $p < i \leq 2p$,

   - Where $\mathbf{x}_F^{(k)}$ is the kth row of X_F, the rows $\mathbf{a}_i$ are fit sequentially by Newton-Raphson iterations with

$$f\left(\mathbf{a}_i\right) \approx \frac{1}{K'} \sum_{k=1}^{K'} \mu \left(\mathbf{a}_i^T \begin{bmatrix} \mathbf{x}_F^{(k)} \\ 1 \end{bmatrix} \right) \begin{bmatrix} \mathbf{x}_F^{(k)} \\ 1 \end{bmatrix}$$

and

$$Hf\left(\mathbf{a}_i\right) \approx \frac{1}{K'} \sum_{k=1}^{K'} \mu'\left(\mathbf{a}_i^T \begin{bmatrix} \mathbf{x}_F^{(k)} \\ 1 \end{bmatrix}\right) \begin{bmatrix} \mathbf{x}_F^{(k)} \\ 1 \end{bmatrix} \begin{bmatrix} [x_F^{(k)}]^T & 1 \end{bmatrix}$$

- If the iterations won't converge, I attempt to fit $f(\mathbf{a}_i) = m_i^*(\tau)$ instead for increasing values of τ until it converges.
- After $\mathbf{a}_i$ is fit, a new column X_i is drawn as independent Bernoulli with probability vector $\mu(X_F \mathbf{a}_i)$.
- X_F is updated to

$$X_F = [X_F \; X_i]$$

3. At the end, the first n rows of X_F are taken as $[X \; \tilde{X}]$, though the rows corresponding to any copy of X would work equally well, since these all should have the desired distribution.

Some thoughts/concerns/explanations:

- Since $x_{-i}^{(k)}$ is replaced with $x_F^{(k)}$, which is a hybrid of real and simulated data, I am not sure there is the same theoretical guarantee that a unique A matrix exists. In practice though it still worked pretty well, and has the advantage that we are deriving $\tilde{X}$ such that $\mathrm{E}(X'\tilde{X}) = \hat{\Sigma} - \mathrm{diag}\{\mathbf{s}\}$.
- A hybrid of simulating $x_{-i}^{(k)}$ and keeping fixed $x_F^{(k)}$ would be to draw from the rows of X K times, with replacement, then simulating the rest of the $x_{-i}^{(k)}$ vector. I'm not convinced there is a good reason to do this.
- By not simulating, there is the advantage of not needing to fit the upper half of A.
- By not redrawing $x_{-i}^{(k)}$ each time, there is less computation for the computer. As well, the multiplication

$$\begin{bmatrix} \mathbf{x}_F^{(k)} \\ 1 \end{bmatrix} \begin{bmatrix} [x_F^{(k)}]^T & 1 \end{bmatrix}$$

needn't be redone for each iteration, and only partially recalculated for each i (though I don't have this implemented yet). The downside might be that error is getting compounded over each i.

4 Bernoulli Knockoff Performance

For each of thses, exaimine these situatations:

- X arises out of the assumed Isling model, with no high order interactions.
- X does not arise out of this model, and does have higher order interactions. Could possibly model this by drawing the vector $\mathbf{x}$ or subsets of it as multinomial with probabilities from a Dirichlet distribution.
- X is from a real world data set, likely something in genetics.
- y is normally distributed around $X\beta$.

- y is drawn from some other distribution, possibly skewed, heavy tailed, or light tailed.
- y is from a real world data set.

4.1 Gram Matrix Comparison

4.2 Binary vs. Determinist Knockoffs in LASSO

4.3 Binary Knockoffs in other Regularlized GLMS

5 Discussion

5.1 Areas for Further Work

6 Further Work and Simulations

As I see it, further simulation work for my paper breaks down into three logical groups: comparison of random Bernoulli knockoffs to the original Knockoffs in LASSO, evaluation of Bernoulli knockoffs in other L1 regularized GLMs, and expanding random knockoffs to more general sorts of variables.

6.1 Comparison to Original Knockoffs in LASSO

The basic idea is to compare how the two sorts of knockoffs compare when X is binary and we are fitting a LASSO regression. Do they select the same variables? Do the binary knockoffs control FDR more or less conservatively? Situations to test:

- X arises out of the assumed Isling model, with no high order interactions.
- X does not arise out of this model, and does have higher order interactions. Could possibly model this by drawing the vector $\mathbf{x}$ or subsets of it as multinomial with probabilities from a Dirichlet distribution.
- X is from a real world data set, likely something in genetics.
- y is normally distributed around $X\beta$.
- y is drawn from some other distribution, possibly skewed, heavy tailed, or light tailed.
- y is from a real world data set.

6.2 Evaluation of Binary Knockoffs in other GLMS

The one of the most interest would be logits. It would be good to see how successful the binary knockoffs are in controlling FDR without the theoretical guarantees that LASSO provides.

- X arises out of the Isling model, in which case it seems like the knockoffs should control FDR.

- X has higher order interactions, which might make the knockoffs perform poorly.
- X is from a real world data set, likely something in genetics.
- y is simulated based on the assumptions of the regression model.
- y is simulated to violate assumptions of the regression model.
- y is from a real world data set.

6.3 Generalizations/Harebrained Ideas

I have two seeds of ideas for extensions if I have enough time.

1. Still with binary data, one might be able to simulate binary variables with higher order interactions in the generation of X by including higher order interactions in the regression of X_i on X_{-i}. This would lead to A being a matrix of higher dimension. Even if this worked, extending the method to knockoffs may not be obvious.

2. In the general case where X is not binary, I can almost imagine a method set up along similar lines to the binary case.

    - The desired covariance matrix could be chosen as in the original knockoff paper.
    - Each x_i's marginal distribution would be approximated by a kernel density estimate on X_i.
    - A $x_i \,\big|\, x_1, \ldots, x_{i-1}$ would be drawn from some reweighing of this marginal to achieve the proper covariance. For instance, if F_i^{-1} is the inverse CDF of the marginal kernel density for x_i and $u_i \,\big|\, x_1, \ldots, x_{i-1}$ is a RV on $(0, 1)$, then $x_i = F_i^{-1}(u_i)$.
    - Maybe fit generalized additive model for each successive x_i with kernel regression for $x_1, \ldots, x_{i-1}$ to predict mean. Then, skew marginal of x_i until it has that mean.