Binary Knockoffs Notes

Aaron Maurer

June 19, 2015

1 Preliminaries

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. x_i will be the random variable corresponding to the ith entry of $\mathbf{x}$, while X_i is the vector of observations drawn from x_i in the data matrix. $\hat{\Sigma} = \frac{1}{n} X^T X$ 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, $\mathrm{diag}(A)$ if the vector of values along the diagonal, while for a vector $\mathbf{a}$, $\mathrm{diag}(\mathbf{a})$ is a square diagonal matrix with $\mathbf{a}$ along the diagonal.

Some early investigation into deterministic knockoffs (as described in the original knockoff paper) 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 $\Sigma \succeq 0$, $\tilde{x}$ is not normally distributed. This can be seen from producing qq plots of X_i vs $\tilde{X}_i$ for each i. Of course, when X is a binary vector, $\tilde{X}$ completely doesn't match its distribution, causing the original X to beat the knockoffs into the model. This indicates that a new method of generating $\tilde{X}$ must be created to control FDR via knockoffs with regularized logistic regression.

2 Random Bernoulli Knockoffs

My idea is to generate $\tilde{\mathbf{x}}$ randomly such that, approximately, $\tilde{\mathbf{x}} \sim \mathbf{x}$. In particular, both variables 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 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 $\mathrm{E}(\mathbf{x}) = \mathbf{m} \in \mathbf{(0,1)^p}$ and cross-moment matrix $\mathrm{E}(\mathbf{xx^T}) = \mathbf{M} \in \mathbf{(0,1)^{p\times p}}$. 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 $i \neq j$[1]. Given a qualifying M, or observed X, there are a few ways of generating more random $\mathbf{x}$.

3.1 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[2][3], but it turns out that there is only certain to exist a working Σ in the bivariate case.

3.2 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 $\mathrm{E}(\mathbf{xx}^T) = M$ when generated from this model[4]. 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.

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

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

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

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

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, 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

$$\begin{aligned}
\mathbf{m}_i &= \mathrm{E}\left(x_i \begin{bmatrix} \mathbf{x}_{-i} \\ x_i \end{bmatrix} \right) \\
\mathbf{m}_i &= \mathrm{E}\left(\mathrm{E}\left(x_i \begin{bmatrix} \mathbf{x}_{-i} \\ x_i \end{bmatrix} \,\middle|\, \mathbf{x}_{-i} \right) \right) \\
\mathbf{m}_i &= \sum_{\mathbf{x}_{-i} \in \{0,1\}^{i-1}} \mathrm{P}(\mathbf{x}_{-i}) \mathrm{P}(x_i = 1 \mid \mathbf{x}_{-i}) \begin{bmatrix} \mathbf{x}_{-i} \\ 1 \end{bmatrix} \\
\mathbf{m}_i &= \sum_{\mathbf{x}_{-i} \in \{0,1\}^{i-1}} \frac{1}{c} \mu\left(\mathbf{x}_{-i}^T A_{-i} \mathbf{x}_{-i}\right) \mu\left(\mathbf{a}_i^T \begin{bmatrix} \mathbf{x}_{-i} \\ 1 \end{bmatrix} \right) \begin{bmatrix} \mathbf{x}_{-i} \\ 1 \end{bmatrix}
\end{aligned}$$

   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

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

   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

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

   and

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

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 $\tau \in [0,1]$

$$\mathbf{m}_i^*(\tau) = (1-\tau)\mathbf{m}_i + \tau \left[\begin{array}{cccc} 0 & \dots & 0 & M_{ii} \end{array} \right]^T$$

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

$$\mathbf{a}_i = \left[\begin{array}{cccc} 0 & \dots & 0 & \log\left(\frac{M_{ii}}{1-M_{ii}}\right) \end{array} \right]^T$$

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

4 Generating 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

   $$\Sigma_L = \left[\begin{array}{cc} \hat{\Sigma} & \hat{\Sigma} - \text{diag}(\mathbf{s}) \\ \hat{\Sigma} - \text{diag}(\mathbf{s}) & \hat{\Sigma} \end{array} \right] \succeq 0$$

   I use a subscript L for large to denote items associated with the joint distribution of $[\mathbf{x}\ \tilde{\mathbf{x}}]$. If $\mathbf{m}_L = \mathrm{E}\left([\mathbf{x}\ \tilde{\mathbf{x}}]\right)^T = [\mathbf{m}\ \mathbf{m}]^T$, 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 $\tilde{x}_i \mid \mathbf{x}, \tilde{x}_1, \dots, \tilde{x}_{i-1}$ so as to create $\tilde{X} \mid X$.

4.1 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.

5 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.

5.1 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.

5.2 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.

5.3 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 $x_i \mid x_1, \ldots, x_{i-1}$ 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 $u_i \mid x_1, \ldots, x_{i-1}$ 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.