STATS 300C: Theory of Statistics **Spring 2022**

Lecture 11 — May 3, 2022

Lecturer: Prof. Emmanuel Candès *Editor: Parth Nobel, Scribe: Nikhil Devanathan*

Warning: *These notes may contain factual and/or typographic errors. They are based on Emmanuel Candès's course from 2018 and 2022, and scribe notes written by Emmanuel Candès, Chenyang Zhong, and Junting Duan.*

Agenda: Controlled Variable Selection

In the previous lecture, we defined the knockoff procedure and proved that Model-X knockoffs are a powerful tool for controlling the FDR. In this lecture, we will explore the construction of Knockoffs and observe their performance in practice.

11.1 Knockoff construction

11.1.1 Conditions

Let $X = \{X_1, \dots, X_n\}$ be a random variable such that $X \sim P_X$ for a known distribution P_X. Let X_{-i} denote $(X_1, \dots, X_{i-1}, X_{i+1}, \dots, X_p)$. We desire the knockoffs $\tilde{X} = \left(\tilde{X}_1, \dots, \tilde{X}_n\right)$ of X to have two key properties:

1. Pairwise exchangeability: for any $S \subseteq \{1, \dots, n\}$,

$$(X, \tilde{X})_{\mathrm{swap}(S)} \overset{d}{=} (X, \tilde{X})$$

2. Conditional independence:

$$\tilde{X}_j \perp\!\!\!\perp Y \mid X_{-j}$$

In the following subsections, we will observe how we can construct knockoffs for various distributions P_X.

11.1.2 Gaussian Case

Making knockoffs when P_X is Gaussian is simple, and we can do it as follows. Let $X = (X_1, X_2) \sim \mathcal{N}(\mathbf{0}, \Sigma)$ where

$$\mathbf{0} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \quad \Sigma = \begin{bmatrix} \sigma_{11} & \sigma_{12} \\ \sigma_{12} & \sigma_{22} \end{bmatrix}.$$

Let $\tilde{X}$ be our knockoff such that $\tilde{X} = (\tilde{X}_1, \tilde{X}_2)$. Let $(X, \tilde{X}) \in P$. By pairwise exchangeability, we have that

$$(X_1, X_2, \tilde{X}_1, \tilde{X}_2) \overset{d}{=} (\tilde{X}_1, \tilde{X}_2, X_1, X_2)$$

and thus,

$$(X_1, X_2) \overset{d}{=} (\tilde{X}_1, \tilde{X}_2).$$

Thus, $P = \mathcal{N}(\mathbf{0}, G)$, where here $\mathbf{0}$ is the zeroes vector of length 4 and G is the following covariance matrix:

$$G = \begin{bmatrix} \Sigma & * \\ * & \Sigma \end{bmatrix}.$$

Using pairwise exchangeability again, we can observe that

$$(X_1, \tilde{X}_2, \tilde{X}_1, X_2) \overset{d}{=} (X_1, X_2, \tilde{X}_1, \tilde{X}_2)$$

so,

$$(X_1, \tilde{X}_2) \overset{d}{=} (X_1, X_2) \overset{d}{=} (\tilde{X}_1, X_2).$$

We can thus deduce that

$$G = \begin{bmatrix} \sigma_{11} & \sigma_{12} & * & \sigma_{12} \\ \sigma_{12} & \sigma_{22} & \sigma_{12} & * \\ * & \sigma_{12} & \sigma_{11} & \sigma_{12} \\ \sigma_{12} & * & \sigma_{12} & \sigma_{22} \end{bmatrix} = \begin{bmatrix} \Sigma & \Sigma - \text{diag}(s) \\ \Sigma - \text{diag}(s) & \Sigma \end{bmatrix}$$

where s is a 2-vector. Ideally, we would like the $*$ elements of G to be 0. We would then have that X_1 and $\tilde{X}_1$ are independent, and X_2 and $\tilde{X}_2$ are independent. Although this is sometimes possible, it is not always the case. Structurally, we know that G must be a symmetric, positive definite matrix to be a covariance matrix. As such, $G_{13} = G_{31}$ and $G_{24} = 42$. If both are 0, depending on $\sigma_{11}, \sigma_{12}, \sigma_{22}$, G may no longer be positive definite. That said, we can minimize any convex function of G_{13} and G_{24} (examples being sum of absolute values or sum of squares) under the constraint that G is positive definite matrix, as the problem is a semidefinite program and thus very feasible to solve with a computer. The most common example is to select $\text{argmax}_s \sum_{i=1}^{2} s_i$ under the constraint that G is positive semidefinite. Once we have selected a G after solving the semidefinite program, we can easily determine the distribution of $\tilde{X} \mid X$. Given n observations $X^{(1)}, X^{(2)}, \ldots, X^{(n)}$ of X, we can generate samples of $\tilde{X}$ from the distribution of $\tilde{X} \mid X$, which we determine from G.

Although we describe the case where X is a length 2 Gaussian vector, this method scales exactly to the case where X is a length p Gaussian vector for integer $p \geq 1$. We use pairwise exchangeability and the covariance matrix Σ of X to populate G, and we set the remaining elements by solving a semidefinite program.

In the following, we show some simulation results in different models [1].

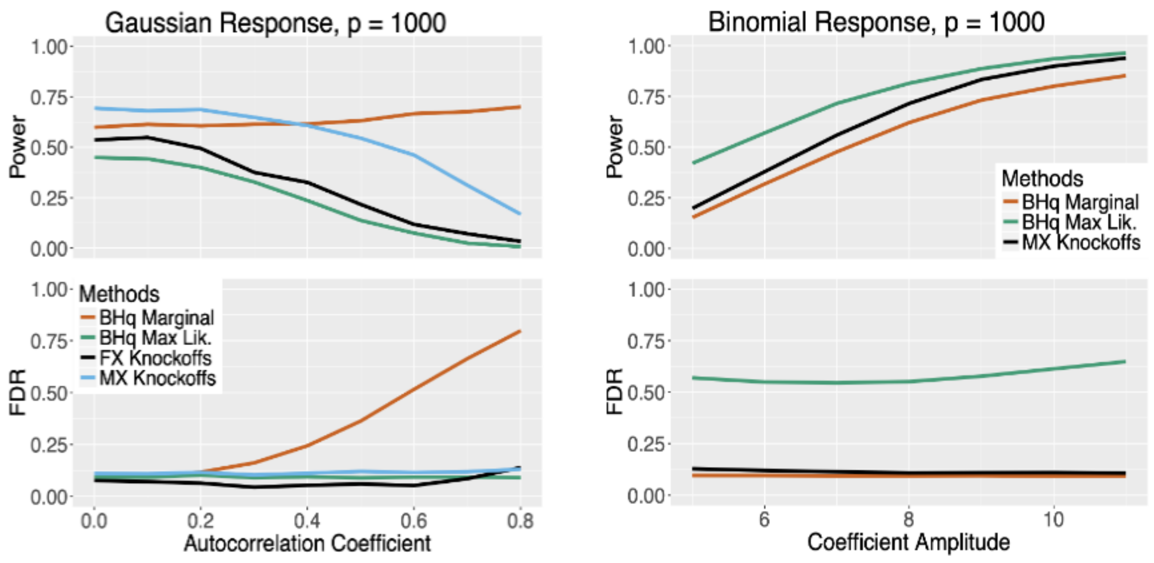

Figure 11.1. Low-dimensional setting: $n = 3000, p = 1000$

- *Low-dimensional linear model with dependent covariates*

  Think of covariate X sampled from an autoregressive model $AR(\rho)$ with varying autocorrelation coefficient ρ. The left-hand side of Figure 11.1 shows the power and FDR of four different methods. We can see that BHq with marginal testing quickly loses FDR control with increasing covariate dependence. This is because the marginal tests are testing the null hypothesis of marginal independence between covariate and response, but all conditionally independent covariates are considered null even if they are marginally dependent on the response. We can also see that Model-X knockoffs is considerably more powerful than the alternatives, since we can use feature important statistics which are much better than ordinary least squares.

- *Low-dimensional logistic model with independent covariates*

  Next we move to a binomial linear model with logit link function and independent covariates. The right-hand side of Figure 11.1 presents the simulation results. Observe that BHq applied to the asymptotic maximum likelihood p-values has an FDR above 50%. Moreover, due to the independence between covariates, there is no difference between marginal testing and conditional testing, so BHq with marginal testing could control the FDR in this setup. And Model-X knockoffs still have more power.

- *High-dimensional logistic model with dependent covariates*

  We also take a look at high-dimensional logistic model in which $n = 3000$ and $p = 6000$. Figure 11.2 shows that BHq with marginal testing does not have FDR control as we observed before.

- *Bayesian knockoff statistics*

  An interesting source of knockoff statistics comes from Bayesian procedures. The statis-

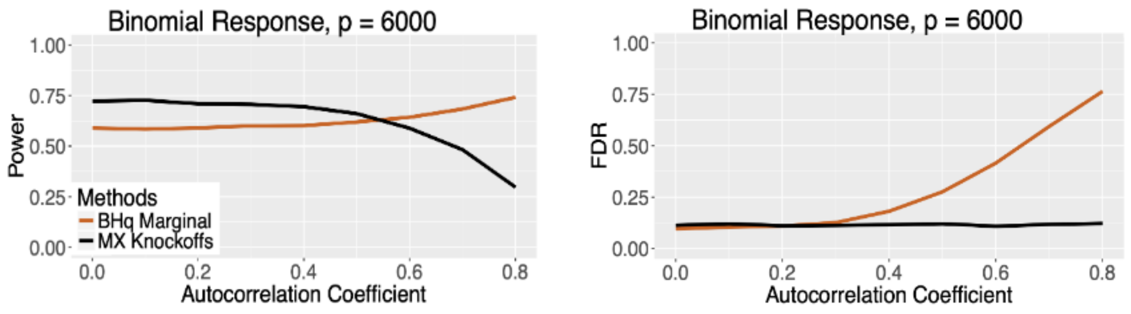

Figure 11.2. Low-dimensional setting: $n = 3000, p = 6000$

tics we used here are the Lasso coefficient-difference (LCD) statistic and a Bayesian variable selection (BVS) statistic, namely $W_j = Z_j - \tilde{Z}_j$ where Z_j and $\tilde{Z}_j$ are the posterior probabilities that the jth original and knockoff coefficients are nonzero respectively. Figure 11.3 shows that if you have a good prior supplied to BVS, it will have more power than LCD which lacks such information. Both of them have FDR control.

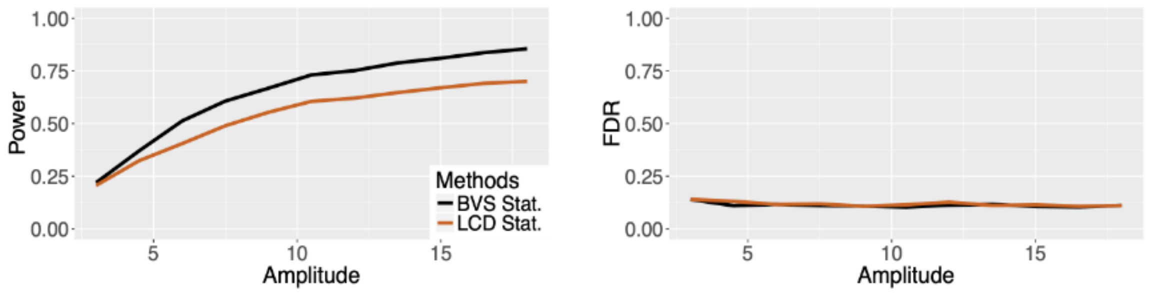

Figure 11.3. $n = 3000, p = 1000$ and Gaussian linear model with 60 expected variables

11.1.3 General Construction

In the general case, (Sesia, Sabatti, Candès (2018)) [5] presents an algorithm for generating samples of the knockoffs given samples of the knockoff $\tilde{X}$ given samples of X and the distribution of X. Let $X = (X_1, \ldots, X_p)$ and $X \sim P_X$ where P_X is known.

Algorithm 1 Sequential Conditional Independent Pairs (SCIP)

```
for j={1,...,p} do
    Sample $\tilde{X}_j$ from law of $X_j \mid X_{-j}, \tilde{X}_{1:j-1}$
end for
```

Through SCIP, the joint law of $(X, \tilde{X})$ is known and and can be proven exchangeable by induction. Although this method allows for sampling knockoffs from arbitrary known distributions, it may be infeasible to compute. Consider the example where

$$\mathbb{P}(X) = e^{-\beta \sum_{i \sim j} X_i X_j}.$$

where $X_i = \pm 1$. In this case, the algorithm is infeasible to compute if p is large. Instead, if X is described by a Markov or hidden Markov model, then SCIP is much easier to compute, and we will see that it runs in linear time.

11.1.4 Knockoffs for Markov Models

Let $X = (X_1, \ldots, X_p) \sim \mathrm{MC}(q_1, Q)$ be a Markov chain. Using the same notation as the previous subsection, the Markov property dictates that $X_i \mid X_{-i} = X_i \mid X_{i-1}, X_{i+1}$. It follows by the definition of conditional probability that the probability density of X is

$$p(X_1, \ldots, X_p) = q_1(X_1) \prod_{j=2}^{p} Q_j(X_j \mid X_{j-1}).$$

In this case, we can implement SCIP efficiently. As depicted pictorially in Figure 11.4, we consider the $p = 4$ case, although this procedure will hold for all integers $p \geq 1$. Let $\mathcal{X}$ be the state space of X. We first sample $\tilde{X}_1$ from $X_1 \mid X_{-1}$. It follows by Bayes's theorem and the Markov property that

$$p(\tilde{X}_1 = \tilde{x}_{-1} \mid X_1 = x_{-1}) \propto q_1(\tilde{x}_1) q Q_2(X_2 \mid \tilde{x}_1).$$

We can calculate the normalizing constant as $\mathcal{N}_1(k) = \sum_{l \in \mathcal{X}} q_1(l) Q_2(k \mid l)$. We can now sample $\tilde{X}_2$ from $X_2 \mid X_1, X_3, \tilde{X}_1$ as

$$p(\tilde{X}_2 = \tilde{x}_2 \mid X_{-2} = x_{-2}, \tilde{X}_1 = \tilde{x}_1) \propto Q_2(\tilde{x}_2 \mid x_1) Q_3(x_3 \mid \tilde{x}_2) \frac{Q_2(\tilde{x}_2 \mid \tilde{x}_1)}{\mathcal{N}_1(\tilde{x}_2)}.$$

It follows that the normalization constant is $\mathcal{N}_2(k) = \sum_{l \in \mathcal{X}} Q_2(l \mid x_1) Q_3(x_3 \mid l) \frac{Q_2(l \mid \tilde{x}_1)}{\mathcal{N}_1(l)}$. This pattern continues.

$$p(\tilde{X}_3 = \tilde{x}_3 \mid X_{-3} = x_{-3}, \tilde{X}_1 = \tilde{x}_1, \tilde{X}_2 = \tilde{x}_2) \propto Q_3(\tilde{x}_3 \mid x_2) Q_4(x_4 \mid \tilde{x}_3) \frac{Q_3(\tilde{x}_3 \mid \tilde{x}_2)}{\mathcal{N}_2(\tilde{x}_3)}$$

$$\mathcal{N}_3(k) = \sum_{l \in \mathcal{X}} Q_3(l \mid x_2) Q_4(x_4 \mid l) \frac{Q_3(l \mid \tilde{x}_2)}{\mathcal{N}_2(l)}$$

$$p(\tilde{X}_4 = \tilde{x}_4 \mid X_{-4} = x_{-4}, \tilde{X}_1 = \tilde{x}_1, \tilde{X}_2 = \tilde{x}_2, \tilde{X}_3 = \tilde{x}_3) \propto Q_4(\tilde{x}_4 \mid x_3) \frac{Q_4(\tilde{x}_4 \mid \tilde{x}_3)}{\mathcal{N}_3(\tilde{x}_4)}$$

$$\mathcal{N}_4(k) = \sum_{l \in \mathcal{X}} Q_4(l \mid x_3) \frac{Q_4(l \mid \tilde{x}_3)}{\mathcal{N}_3(l)}$$

11.1.5 Knockoffs for Hidden Markov Models

Knockoffs are similarly straightforward to sample for hidden Markov models (HMM). $X = (X_1, \ldots, X_p)$ an HMM if:

1. $H \sim \mathrm{MC}(q_1; Q)$ (latent Markov chain)

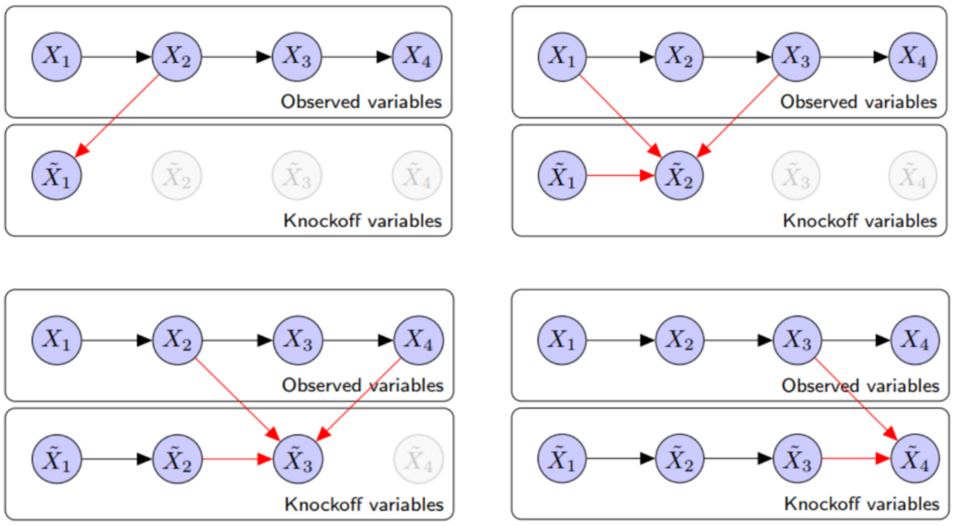

Figure 11.4. Knockoff construction for Markov chain

2. $X_j \mid H \sim X_j \mid H_j \overset{ind.}{sim} f_j(X_j; H - j)$ (emission distribution).

In an HMM, the latent Markov chain H is unobservable, and we instead observe only the emission distribution X. An HMM with $p = 3$ is illustrated in Figure 11.5.

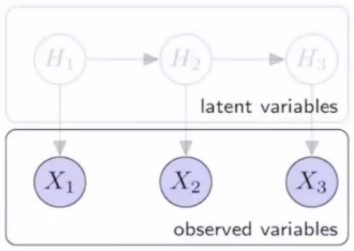

Figure 11.5. Hidden Markov model ($p = 3$)

If we have vector X of observations from an HMM, we can sample knockoffs $\tilde{X}$ using the following procedure from (Sesia, Sabatti, Candès (2018)) [5]:

1. Sample H from $p(H \mid X)$ using forward-backward algorithm
2. Generate a knockoff $\tilde{H}$ of H using the SCIP algorithm for a Markov chain
3. Sample $\tilde{X}$ from the emission distribution of X given $H = \tilde{H}$

This logic is presented in Figure 11.6.

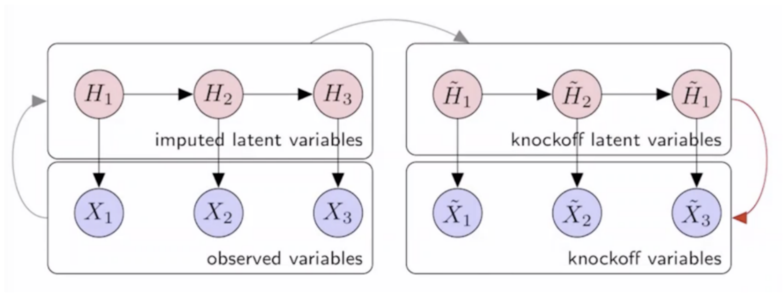

Figure 11.6. Knockoff construction for hidden Markov model

Hidden Markov models are widely used as a phenomenological model for haplotype estimation and phasing, and imputation of missing SNPs. Each genotype can be modeled as the sum of two independent HMM haplotype sequences. Though we do not know exact model parameters, we can estimate them from data, for example use software fastPHASE. In GWAS study, the variables are genotype for a set of SNPs and the response can be status of a disease or a quantitative trait of interest. With the above procedure, we can generate approximate knockoff copies of genotype sequences and use them to make inference.

Figure 11.7 and 11.8 show the power and FDP of this procedure using true parameters and estimated parameters of the HMM covariates, respectively. There is almost no difference between these two sets of plots, showing the robustness of this procedure.

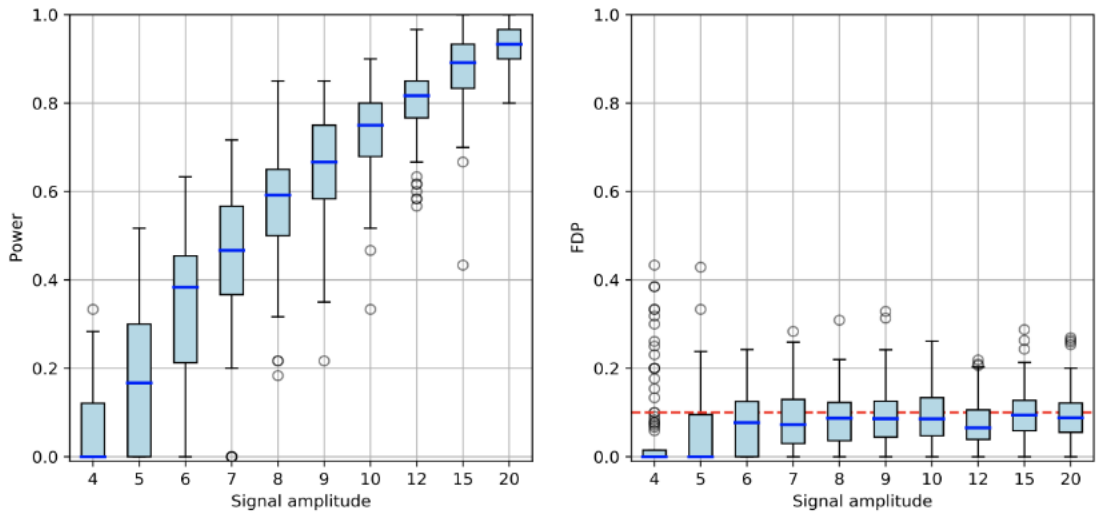

Figure 11.7. Power and FDP over 100 repetitions (true F_X). $n = 1000, p = 1000$, target FDR:$\alpha = 0.1, Z_j = |\hat{\beta}(\hat{\lambda}_{\mathrm{CV}})|, W_j = Z_j - \tilde{Z}_j$.

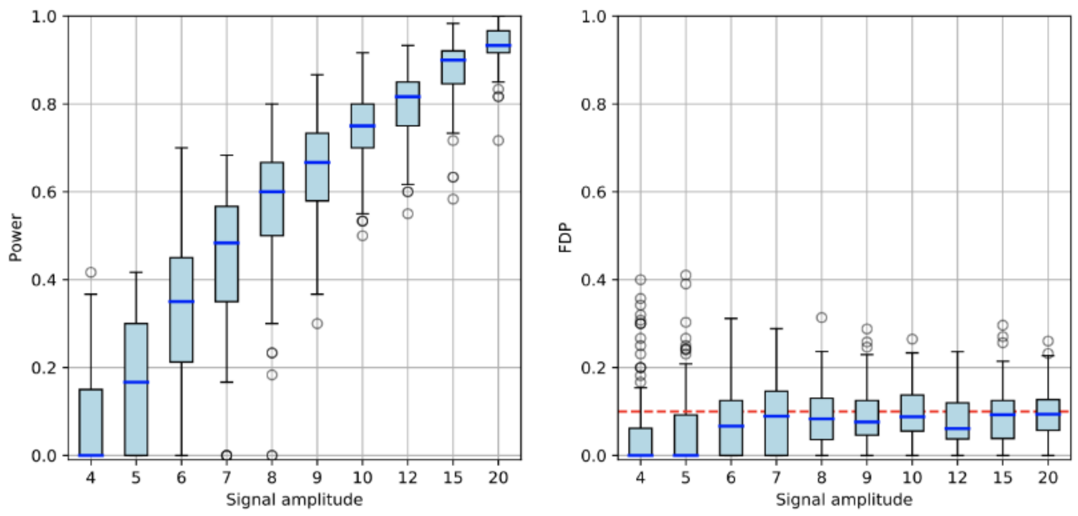

Figure 11.8. Power and FDP over 100 repetitions (estimated F_X). $n = 1000, p = 1000,$ target FDR: $\alpha = 0.1, Z_j = |\hat{\beta}(\hat{\lambda}_{\text{CV}})|, W_j = Z_j - \tilde{Z}_j$.

11.2 Application: Knockoffs for UK Biobank

There are some applications of knockoff method in genetics:

- Sesia, Katsevich, Bates, Candès, Sabatti (2020) ”Multi-resolution localization of causal variants across the genome,” Nature Communications [7]
- Sesia, Bates, Candès, Marchini, Sabatti (2021) ”Controlling the false discovery rate in GWAS with population structure” [6]

The paper takes a look at the UK Biobank dataset in which 592,000 single nucleotide polymorphisms (p) are genotyped on a complicated population of 489,000 individuals (n). The goal is to discover genetic variants that influence various traits. After producing the knockoffs, we should expect them exhibit exactly the same structure as the original data since $\tilde{X}$ is distributed as X. Specifically, the knockoffs should preserve population structure (see in Figure 11.9), relatedness, cross correlations (linkage disequilibrium), etc.

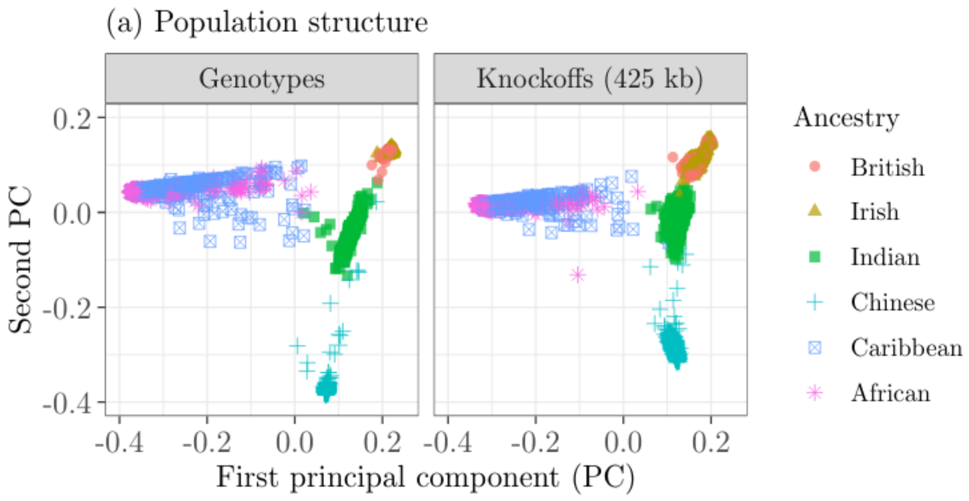

Figure 11.9.

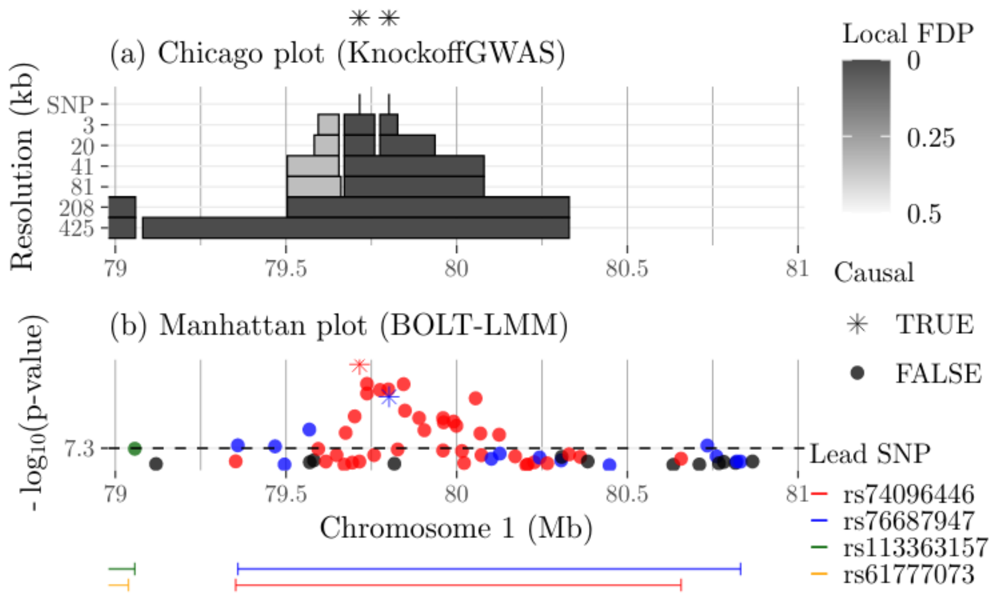

Figure 11.10.

Since the nearby location of genome would be highly correlated, it's difficult to tell which one is important. A possible solution is to group nearby variables as a set G and look at a simpler problem: whether Y is conditional independent of G given everything else, i.e. $Y \perp\!\!\!\perp X_G \mid X_{-G}$. Hence, we could have a knockoff discovery scheme that runs at various resolutions: from the smallest resolution to the highest resolution (individual snips), which is illustrated in Figure 11.10.

An advantage of this method is that, once we have our knockoffs, we can use any model without making any assumption on the structure of it. If the knockoffs are good, the method is robust to everything else.

11.3 Deep Knockoffs

Thus far, we have covered methods for constructing/sampling from knockoffs of X in the case that the distribution of X is known. More recently novel research has been conducted on generating knockoffs of X when the distribution of X is unknown using neural networks [3] [4].

11.3.1 Training a Neural Net to Create Knockoffs

Suppose we have some random variable $X = (X_1, \dots, X_p)$ with unknown distribution P_X and some observations $X^1, \dots, X^n$ of X. We can create a deep neural net f_θ an observation of X along with some Gaussian noise to return a new "observation" of X. Ideally, we want to train neural net to output a suitable knockoff observation of $\tilde{X}$ given a true observation of X.

In order to be a useful knockoff, $\tilde{X}$ must generally be pairwise exchangeable, so we can roughly compute the distance to pairwise exchangeability of X and $\tilde{X}$ as

$$J_\theta(X, \tilde{X}) = \sum_{j=1}^{p} \mathcal{D}\left((X, \tilde{X}), (X, \tilde{X})_{\text{swap}(j)}\right) + \delta \sum_{j=1}^{p} (X^\top \tilde{X})^2$$

which we compute using samples of X and corresponding samples of $\tilde{X}$. We can update the deep neural net using stochastic gradient descent and update θ based on $\nabla_\theta J_\theta(X, \tilde{X})$. Once the neural net is trained, we can run knockoff procedures using its outputs.

11.3.2 Maximum Mean Discrepancy

Given two random variables U and V (such as an RV and its deep knockoff), we want to test whether $P_U = P_V$. We can determine the discrepancy between the two distributions by considering the distances between embeddings of the distributions to reproducing kernel Hilbert spaces (RKHS) [2]. Let $\phi(U)$ and $\phi(V)$ be the embedding of U and V. We can compute the discrepancy to be

$$\mathcal{D}_\phi = \|\mathbb{E}_U[\phi(U)] - \mathbb{E}_V[\phi(V)]\|_{\mathcal{H}}^2.$$

If we let $\phi(U) = U$ and $\phi(V) = V$, then the MMD gives the distance between means. If we let $\phi(U) = (U, U^2)$ and $\phi(V) = (V, V^2)$, then the MMD gives the error between the first two moments.

To compare higher order moments, we use the "kernel trick". We expand the quadratic and replace inner products with kernel operations. We define the maximum mean discrepancy (MMD) such that

$$\text{MMD}(P_U, P_V) = \mathbb{E}_{UU'}[\kappa(U, U')] - 2\mathbb{E}_{UV}[\kappa(U, V)] + \mathbb{E}_{VV'}[\kappa(V, V')]$$

Using kernel operators, we ensure that $P_U = P_V$ if and only if the MMD of U and V is 0. Given n observations $U^1, \dots, U^n$ and $V^1, \dots, V^n$ of U and V respectively, we can calculate an unbiased estimator of the MMD as

$$\widehat{\text{MMD}} = \frac{1}{n(n-1)} \sum_{i=1}^{n} \sum_{j \neq i} \kappa(U^i, U^j) - \frac{2}{n^2} \sum_{i=1}^{n} \sum_{j=1}^{n} \kappa(U^i, V^j) + \frac{1}{n(n-1)} \sum_{i=1}^{n} \sum_{j \neq i} \kappa(V^i, V^j).$$

Utilizing the MMD, we get a procedure by which we can train a neural net to generate knockoffs. Let X^i for $i = 1, \ldots, n$ be an observation of $X \sim P_X$ where P_X is unknown, let Z^i for $i = 1, \ldots, n$ be an observation of $Z \sim \mathcal{N}(0, I)$, let f_{θ_t} be a neural net with parameters θ_t (tth iteration parameters).

1. We generate knockoff samples $\tilde{X}^i$ such that $\tilde{X}^i = f_{\theta_t}(X^i, Z^i)$ for $i = 1, \ldots, n$. In this step f_{θ_t} is fixed.

2. Next we evaluate the loss

$$J_{\theta_t}(X_t, \tilde{X}_t) = \sum_{j=1}^{p} \widehat{\text{MMD}} \left((X, \tilde{X}), (X, \tilde{X})_{\text{swap}(j)} \right) + \delta \sum_{j=1}^{p} (X_j^\top \tilde{X}_j)^2.$$

3. Finally, we update the parameters $\theta_{t+1} \leftarrow \theta_t - \mu \nabla_{\theta_t} J_{\theta_t}(X_t, \tilde{X}_t)$ and repeat the process.

We can optimize this procedure by using stochastic gradient descent to update the parameters, greatly reducing the number of calculations per iteration. More information regarding using knockoffs can be found on Prof. Candès's group's website.

Bibliography

[1] Emmanuel Candès, Yingying Fan, Lucas Janson, and Jinchi Lv. Panning for gold: Model-x knockoffs for high-dimensional controlled variable selection, 2016.

[2] Arthur Gretton, Karsten M. Borgwardt, Malte J. Rasch, Bernhard Schölkopf, and Alexander Smola. A kernel two-sample test. *Journal of Machine Learning Research*, 13(25):723–773, 2012.

[3] James Jordon, Jinsung Yoon, and Mihaela van der Schaar. KnockoffGAN: Generating knockoffs for feature selection using generative adversarial networks. In *International Conference on Learning Representations*, 2019.

[4] Ying Liu and Cheng Zheng. Deep latent variable models for generating knockoffs. *Stat*, 8(1):e260, 2019. e260 sta4.260.

[5] M Sesia, C Sabatti, and E J Candès. Gene hunting with hidden markov model knockoffs. *Biometrika*, 106(1):1–18, aug 2018.

[6] Matteo Sesia, Stephen Bates, Emmanuel Candès, Jonathan Marchini, and Chiara Sabatti. Fdr control in gwas with population structure. *bioRxiv*, 2021.

[7] Matteo Sesia, Eugene Katsevich, Stephen Bates, Emmanuel Candès, and Chiara Sabatti. Multi-resolution localization of causal variants across the genome. *bioRxiv*, 2019.