ETH zürich

High-Dimensional Inference: Confidence Intervals, *p*-values and R-Software *hdi*

Nick Rüdlinger, Lauro Langosco

About the Paper

- Paper from Seminar of Statistics
  - Title: *High-Dimensional Inference: Confidence Intervals, p-values and R-Software hdi* (2015)
  - Ruben Dezeure (former PhD), Peter Bühlmann (Professor), Lukas Meier (Senior Scientist) and Nicolai Meinshausen (Professor)
- Summary of some other papers about high-dimensional inference

Overview

Introduction

Multi Sample-Split

Desparsified Lasso

Comparison

Hierarchical Clustering

Basic Terminology

Consider a high-dimensional linear model: $Y = \boldsymbol{X}\beta^0 + \varepsilon$ (fixed or random design)

- active set: $S_0 = \{j : \beta_j^0 \neq 0, j = 1, ..., p\}$

Hypothesis testing: $H_{0,j} : \beta_j^0 = 0$, for $j = 1, ..., p$

- Type I error: rejection of a true null hypothesis
- Type II error: failing to reject a false hypothesis

Basic Terminology

- *p*-**value**: how likely our observation or an even more extreme observation (of the test statistics) is, assuming H_0 is true

- **Family-wise Error Rate**: Probability to make one or more false discoveries:

$$P(\exists\, j \in S_0^C : H_{0,j} \text{ is rejected})$$

Restrictions on the design

In high-dimensional setting and for general fixed design, regression parameters are not always identifiable

Compatibility Condition:
$\exists \phi_0 > 0$ s.t. $\forall \beta$ satisfying $||\beta_{S_0^c}||_1 \leq 3||\beta_{S_0}||_1$ it holds:

$$||\beta_{S_0}||_1^2 \leq \frac{|S_0|}{n\phi_0^2} \cdot \beta^T \boldsymbol{X}^T \boldsymbol{X} \beta$$

guarantees identifiability and oracles near optimal results for the Lasso estimator

Restrictions on the Parameters

Another often used, but not necessary assumption, is the so-called **beta-min assumption:**

$$\min_{j \in S_0} |\beta_j^0| \geq \beta_{min}$$

useful: beta-min assumption and compatibility condition imply the screening property for the Lasso

screening property: $\hat{S} = \{j : \hat{\beta}_j \neq 0\} \supseteq S_0$
gives a massive dimensional reduction because $|\hat{S}| \leq \min(n, p)$

Overview

Single sample-split

1. Partition the sample: $\{1, ..., n\} = I_1 \cup I_2$ such that $I_1 \cap I_2 = \emptyset$ and $|I_1| = \lfloor n/2 \rfloor$ and $|I_2| = n - \lfloor n/2 \rfloor$
2. Use I_1 only for variable selection and enforce that $|\hat{S}(I_1)| \leq |I_1| \leq |I_2|$:
   - use Lasso: select variables which have estimated regression coefficient different from zero
3. Reduce the data to $(Y_{I_2}, \boldsymbol{X}_{I_2}^{(\hat{S}(I_1))})$ which is low dimensional: use ordinary least squares estimation (i.e. t-test)

$$P_{raw,\, j} = \begin{cases} P_{t-test,\, j}\,, & \text{if } j \in \hat{S}(I_1) \\ 1\,, & \text{if } j \notin \hat{S}(I_1) \end{cases}$$

Single sample-split: multiple testing adjustment

To control the FWER over all considered hypotheses:

Bonferroni correction: $P_{corr,j} = \min(P_{raw,j} \cdot |\hat{S}(I_1)|, 1)$

The resulting p-values control the FWER if we assume that the screening porperty hold.

Why does this work? Screening property ensures that the reduced model is a correct model.

Single sample-split: *p*-value lottery

Problem: Sensitivity with respect to splitting the sample

R code: pvalue_lottery_ribovlavin.R

Solution: multi sample-split

Run single sample-split multiple (B) times.
$\rightarrow$ collection of p-values for each Hypothesis $H_{0,j}$:

$$P^{[1]}_{corr,j}, ..., P^{[B]}_{corr,j} \quad (j = 1, ..., p)$$

Aggregation to single p-value:

$$P_j = \min((1 - \log(\gamma_{min})) \inf_{\gamma \in (\gamma_{min}, 1)} Q_j(\gamma))$$

where: $Q_j(\gamma) = \min(\text{emp. } \gamma\text{-quantile}\{P^{[b]}_{corr,j}/\gamma\})$

such p-values are approximately reproducible and not subject to p-value lottery anymore

MS-Split: Assumptions

(**A1**) screening property for the first half of the sample I_1:
$P(\hat{S}(I_1) \supseteq S_0) \geq 1 - \delta$ for some $0 < \delta < 1$
(**A2**) the reduced design matrix for the second half of the sample has full rank: $\text{rank}(\boldsymbol{X}_{I_2}^{\hat{S}(I_1)}) = |\hat{S}(I_1)|$

FACT 1

consider a linear model with fixed design $\boldsymbol{X}$ *and Gaussian errors. If (A1) & (A2) hold, then for a significance level* $0 < \alpha < 1$ *and B sample-splits it holds:*

$$P(\bigcup_{j \in S_0^c} \{P_j \leq \alpha\}) \leq \alpha + B\delta$$

that is, the FWER is controlled up to the additional small value $B\delta$

Generality of the MS-Split

The multi sample-split method is very generic

- any sparse method can be used for variable screening as long as (A1) & (A2) hold

- beta-min assumption and identifiability condition imply (A1) & (A2) for many sparse estimators

The Lasso has been empirically found to perform very well compared to other estimators (e.g. adaptive Lasso, elastic net)

Problem with beta-min

$$\min_{j \in S_0} |\beta_j^0| \geq \beta_{min}$$

Is an assumption about the unknown β_0 and the absolute values of its components.

- **very unpleasant**: this is the question of the Hypothesis test!

Confidence Intervals

So far we only got *p*-values. How do we get Confidence Intervals?

Idea: use **duality with the *p*-values!**

if we have a test at level α then a $(1-\alpha)$-CI is given by:

$$\{c \mid p\text{-value} \geq \alpha \text{ for testing } H_{0,j} : \beta_j = c\}$$

Multi sample-split: Repetition

1. split sample in two disjoint halves I_1 & I_2
2. use I_1 for variable selection & enforce that: $|\hat{S}(I_1)| \leq |I_2|$
3. apply OLS and t-test to the reduced low-dim data $(Y_{I_2}, \boldsymbol{X}_{I_2}^{\hat{S}(I_1)})$ to generate p-values
4. Bonferroni correction to adjust for multiple testing $P_{corr,j} = \min(P_{raw,j}|\hat{S}(I_1)|, 1)$
5. repeat single-split (1-4) B times (typically B = 50 or 100)
6. aggregate the B p-values to a single p-value by choosing an empirical γ-quantile

Overview

Desparsified Lasso

Model:

$$Y = X\beta + \varepsilon.$$

Assume we are in a low-dimensional setting, i.e. $p < n$ and X has full rank.

Desparsified Lasso

The j-th component $\hat{\beta}_j$ of the OLS estimate $\hat{\beta}$ can be obtained as follows:

- Do OLS regression of the j-th predictor $X^{(j)}$ vs. all the other predictors $X^{(-j)}$.
- Denote the corresponding residuals by $Z^{(j)}$.
- Then

$$\hat{\beta}_j = \frac{Y^T Z^{(j)}}{(X^{(j)})^T Z^{(j)}}.$$

Note: this does not work if $p > n$.

Desparsified Lasso

Model:

$$Y = X\beta + \varepsilon,$$

where now $p > n$.

Instead of OLS, we now use a Lasso regression of $X^{(j)}$ vs. $X^{(-j)}$. Again denote the residuals by $Z^{(j)}$. The **desparsified Lasso** estimator is then given by

$$\hat{b}_j = \frac{Y^T Z^{(j)}}{(X^{(j)})^T Z^{(j)}} - \text{[bias correction term]}.$$

Desparsified Lasso

Model:

$$Y = X\beta + \varepsilon,$$

where now $p > n$.

Instead of OLS, we now use a Lasso regression of $X^{(j)}$ vs. $X^{(-j)}$. Again denote the residuals by $Z^{(j)}$. The **desparsified Lasso** estimator is then given by

$$\hat{b}_j = \frac{Y^T Z^{(j)}}{(X^{(j)})^T Z^{(j)}} - \frac{1}{(X^{(j)})^T Z^{(j)}} \sum_{k \neq j} (X^{(j)})^T Z^{(j)} \hat{\beta}_k,$$

where $\hat{\beta}$ is the result of a Lasso regression of Y on X.

Desparsified Lasso

Our goal is to construct p-values and confidence intervals.
It turns out that, **asymptotically**,

$$\sqrt{\frac{n}{\sigma^2 \Omega_{jj}}}(\hat{b}_j - \beta_j) \sim \mathcal{N}(0, 1),$$

where

- σ^2 is the error variance,
- $\Omega_{jj} = n \frac{(Z^{(j)})^T Z^{(j)}}{((X^{(j)})^T Z^{(j)})^2}$.

Desparsified Lasso

Model:

$$Y = X\beta + \varepsilon.$$

Desparsified Lasso:

1. Compute Lasso regression $\hat{\beta}$ of Y vs. X.
2. Compute Lasso regression $\hat{\beta}^{(j)}$ of $X^{(j)}$ vs. $X^{(-j)}$ for all $j \in \{1, \dots, p\}$.
3. The **desparsified Lasso** estimator is given by

$$\hat{b}_j = \frac{Y^T Z^{(j)}}{(X^{(j)})^T Z^{(j)}} - \frac{1}{(X^{(j)})^T Z^{(j)}} \sum_{k \neq j} (X^{(j)})^T Z^{(j)} \hat{\beta}_k.$$

4. We can construct asymptotic confidence intervals and p-values for $\hat{b}$.

Overview

Comparison of the methods

Compare the methods based on FWER and Power:

- use empirical estimates of:

$$\text{Power} = \sum_{j \in S_0} P(H_{0,j} \text{is rejected}) / |S_0|$$

$$\text{FWER} = P(\exists\, j \in S_0^C : H_{0,j} \text{ is rejected})$$

- for given $\boldsymbol{X}$ & β generate 100 different responses and apply method for each response $\rightarrow$ 100 p-values $\forall j$
  - count how many times event happens

R Code Example

- **simple_comparison.R**: A quick comparison of the MS-Split and Desparsified Lasso
- **comparison_2.R**: script to generate estimates of the FWER and power
- **data_aggregation.R**: script to load and display the estimates of the FWER and power which are contained in *fwer.rds* and *pow.rds*

Overview

Hierarchical Inference

- The methods presented so far assume that the contributions of individual predictors are large enough to be detected.
- But: for high-dimensional data, predictors are often highly correlated.
- Therefore, confidence intervals can be wide and uninformative.

Hierarchical Inference

- Computing *p*-values for individual variables can be difficult.
- Instead: compute *p*-values and confidence intervals for groups of predictors.

Hierarchical Inference

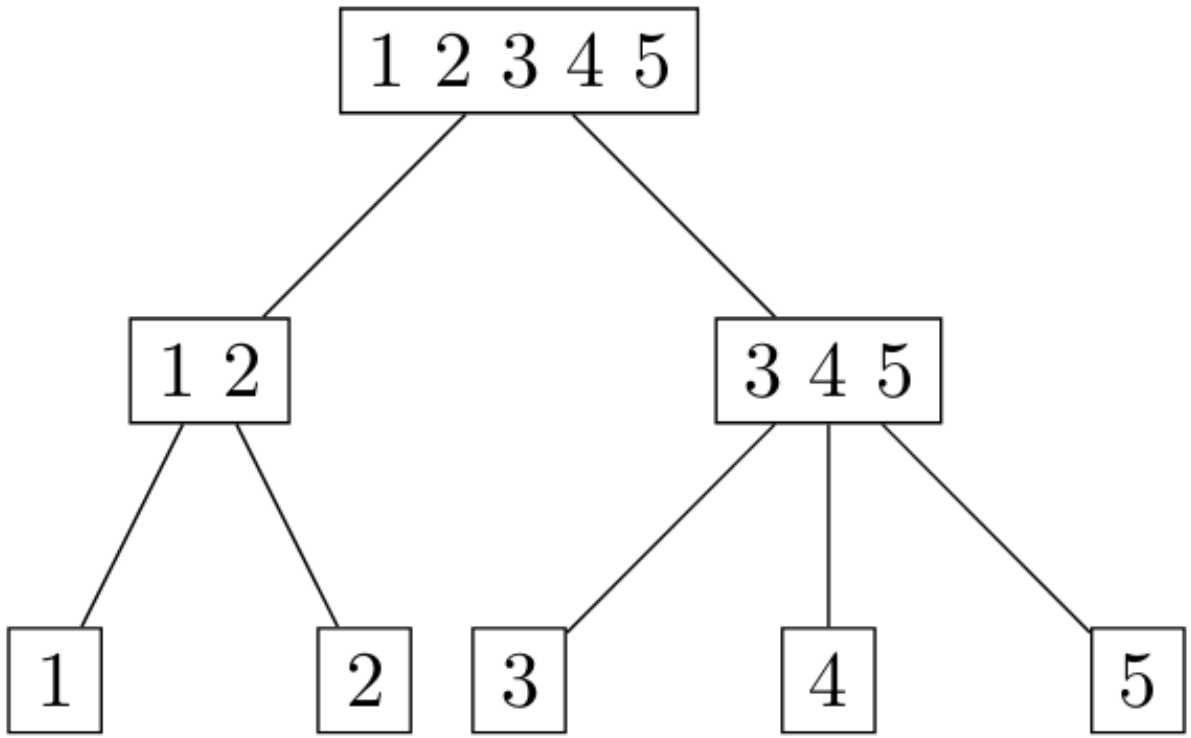

Figure: A hierarchy.

Hierarchical Inference

Definition

A *hierarchy* $\mathcal{T}$ is a set of clusters $\{\mathcal{C}_k; k\}$ with $\mathcal{C}_k \subset \{1, \dots, p\}$ such that

- there is a root node (cluster) that contains all variables $\{1, \dots, p\}$.
- for two clusters $\mathcal{C}_k, \mathcal{C}_l$, either one is a subset of the other or they have an empty intersection.

Usually, there is also an added notion of a level such that, on each level, the clusters form a partition of $\{1, \dots, p\}$.

R Code

Demonstration in R.

- **clustering.R**: Demonstration of hierarchical clustering using `hclust`.

How Do We Get the Clusters?

- To compute the hierarchy, we need a distance measure on the variables.
- Starting with the individual variables, we iteratively merge the clusters closest to each other.
- For this, we need a measure of distance between clusters, e.g. the average distance between variables in two clusters.

Hierarchical Inference

Inference works as follows (given a hierarchy):

- Begin: test the root node cluster $\mathcal{C}_0 = \{1, \dots, p\}$ with the *cluster null hypothesis* $H_{0,\mathcal{C}_0} : \beta_1 = \cdots = \beta_p = 0$.
  - If we fail to reject: we are done (nothing is significant).
  - If we reject: move on to the next level and test the hypotheses $H_{0,\mathcal{C}}$ for all clusters $\mathcal{C}$ on the second level.
- Go on like this until no more cluster hypotheses can be rejected.

Hierarchical Inference

1. Create groups / clusters of predictors (e.g. from correlation structure).
2. Choose a significance test for groups.
3. Test the groups for significance, starting from the full cluster $\{1, \dots, p\}$. Stop when no more cluster null hypotheses can be rejected.

Hierarchical Inference

- We need a method to test cluster null hypotheses $H_{0,\mathcal{C}}$.
- The package `hdi` implements this using a method called the *Group-bound*.
- Group-bound gives confidence intervals for the $\mathcal{L}^1$-norm $\|\beta_{\mathcal{C}}\|_1$ of a group of predictors.

R Code

Demonstration in R.

- **Inference.R** A demonstration of hierarchical inference using `clusterGroupBound` from the `hdi` package.