MSA220/MVE440 Statistical Learning for Big Data

Lecture 9-10 - High-dimensional regression

Rebecka Jörnsten

Mathematical Sciences
University of Gothenburg and Chalmers University of Technology

Recap from last lecture

We want to fit a regression model to our data using least squares.

$$y = X\beta + \epsilon$$

- y is our $n \times 1$ vector of outcome data
- X is our $n \times p$ design matrix
- ϵ is our $n \times 1$ vector with additive errors.
- For convenience, assume centered and standardized data.

Is this OK?
Reality check: 5 basic assumptions, scatter plots,....
TRANSFORMATIONS! ID EXTREME OUTLIERS!

Recap from last lecture

When $n > p$ and no X's are perfectly correlated we can solve the least squares problem directly as

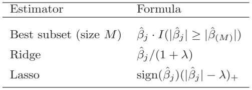
- $\hat{\beta} = (X'X)^{-1}X'y$
- The variance of $\hat{\beta} = \sigma^2(X'X)^{-1}$ where σ^2 is the error variance
- Notice that a main source of variance is $(X'X)^{-1}$
- When p is large or Xs correlated, this inverse becomes unstable (determinant approaches 0) which is reflected in high estimation variance for the regression coefficients.

Regularized regression

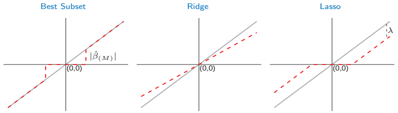
$p > n$ problem: use penalized least squares

$$\frac{1}{2}||y - X\beta||^2 + \lambda J(\beta)$$

- As you saw last lecture the penalty J can be chosen in many ways
- $J(\beta) = ||\beta||_2^2$ (L2) gives us the ridge regression estimates.
- $J(\beta) = ||\beta||_1$ (L1) gives us the lasso estimates
- Both are biased, lasso with a constant bias λ for non-zero estimates whereas ridge has an increasing bias with coefficient magnitude.
- Both are *continuous in data*, essential to not be too sensitive to data perturbations.



Estimator	Formula
Best subset (size M)	$\hat\beta_j \cdot I(\lvert\hat\beta_j\rvert \geq \lvert\hat\beta_{(M)}\rvert)$
Ridge	$\hat\beta_j/(1+\lambda)$
Lasso	$\text{sign}(\hat\beta_j)(\lvert\hat\beta_j\rvert - \lambda)_+$



Regularized regression

$p > n$ problem: use penalized least squares

$$\frac{1}{2}||y - X\beta||^2 + \lambda J(\beta)$$

- There were more complex $J()$ we could consider
- *group-lasso* $J(\beta) = \sum_g^G ||\beta_g||_2$ penalizes a group mean for the coefficients which has the effect of selecting a group of coefficients to be "in" or "out" in the model
- If you add the L1-penalty $||\beta||_1$ to the group penalty you get *sparse group-lasso* which favors groups to enter the model but "cleans up" the coefficients in the group that are not contributing.
- *Elastic net* uses a weighted combination of the L2 and L1 penalties. This has the effect of keeping correlated variables together in the model. You don't have to define the groups.

L1-penalized regression

$$\frac{1}{2}||y - X\beta||^2 + \lambda||\beta||_1$$

Consider first the special case $X'X = I$.

$$\frac{1}{2}y'y - y'X\beta + \frac{1}{2}\beta'\beta + \lambda||\beta||_1 = ***$$

Take derivatives with respect to β_j:

$$\frac{\partial ***}{\partial \beta_j} = -x_j'y + \beta_j + \lambda\nu_j$$

where

$$\nu_j = \begin{cases} sign(\beta_j) & if \quad \beta_j \neq 0 \\ \{\nu_j : |\nu_j| \leq 1\} & if \quad \beta_j = 0 \end{cases} \tag{1}$$

L1-penalized regression

$$\frac{\partial ***}{\partial \beta_j} = -x_j' y + \beta_j + \lambda \nu_j$$

where

$$\nu_j = \begin{cases} sign(\beta_j) & if \quad \beta_j \neq 0 \\ \{\nu_j : |\nu_j| \leq 1\} & if \quad \beta_j = 0 \end{cases} \tag{2}$$

So if $\beta_j > 0$, this is $\hat{\beta}_j = x_j' y - \lambda$ and if $\beta_j < 0$ this is $\hat{\beta}_j = x_j' y + \lambda$. There is a conflict between the assumed sign and the solution if $|x_j' y| < \lambda$. Note, $x_j' y = \hat{\beta}_j^{LS}$ for this special case $X'X = I$.
Solution:

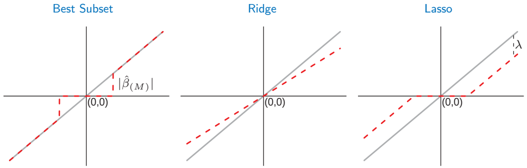
$$\hat{\beta}_j = \begin{cases} \beta_j^{LS} - \lambda & if \quad \beta_j^{LS} > \lambda \\ \beta_j^{LS} + \lambda & if \quad \beta_j^{LS} < -\lambda \\ 0 & if \quad |\beta_j^{LS}| < \lambda \end{cases} \tag{3}$$

This is called the *Soft Thresholding operation*, ST and we write

$$\hat{\beta}_j = ST(x_j' y, \lambda)$$

L1-PENALIZED REGRESSION

Estimator	Formula
Best subset (size M)	$\hat\beta_j \cdot I(\lvert\hat\beta_j\rvert \geq \lvert\hat\beta_{(M)}\rvert)$
Ridge	$\hat\beta_j/(1+\lambda)$
Lasso	$\text{sign}(\hat\beta_j)(\lvert\hat\beta_j\rvert - \lambda)_+$



This figure from the book illustrates the estimation of $L0$, $L1$ and $L2$ in relation to the LS estimate.
Notice the constant bias for $L1$ and growing bias for $L2$.

L1-penalized regression

What about the general case? We can't solve this with a closed-form expression. But there are tons of ways of solving this, numerically and iteratively.
Here I illustrate *coordinate descent* which usually leads to simple steps and can be fast for big problems.

$$\min_{\beta} \frac{1}{2}||y - X\beta||^2 + \lambda||\beta||_1 = **$$

L1-penalized regression

Take derivatives wrt β_j

$$\frac{\partial **}{\partial \beta_j} = -X_j'(y - \sum_{l \neq j} X_l \beta_l - X_j \beta_j) + \lambda \nu_j$$

We can write this as

$$-X_j' r_j + X_j' X_j \beta_j + \lambda \nu_j = 0$$

where r_j is the residual keeping the other $\beta_l, l \neq j$ fixed.
Now we have an expression that looks very much like our special case from before and so the solution is

$$\hat{\beta}_j = \frac{ST(X_j' r_j, \lambda)}{X_j' X_j}$$

If $\lambda = 0$ this is a iterative procedure for estimating LS coefficients which agrees with a simple updating scheme:

$$\hat{\beta}_j = \beta_j^{old} + X_j' r / X_j' X_j$$

L1-penalized regression

Another popular method for solving constrained optimization problems is ADMM; *alternating directions method of multipliers*. This method decomposes the problem into separate convex optimization parts and then regularizes the solution differences. Here's the original problem:

$$\min_{\beta} \frac{1}{2}||y - X\beta||^2 + \lambda||\beta||_1$$

The augmented lagrangian version looks like this:

$$\min_{\beta} \frac{1}{2}||y - X\beta||^2 + \lambda||\theta||_1 + u'(\beta - \theta) + \frac{\rho}{2}||\beta - \theta||^2$$

The lagrangian parameter u controls the differences between the two solutions β and θ and the term with ρ is a tuning term that can help speed up convergence. (Please read S. Boyd's excellent notes and papers on ADMM for more information).

L1-penalized regression

$$\min_{\beta} \frac{1}{2}||y - X\beta||^2 + \lambda||\theta||_1 + u'(\beta - \theta) + \frac{\rho}{2}||\beta - \theta||^2$$

We now solve the problem by first minimizing wrt β, then θ and then updating u as
$u^{t+1} = u^t + \rho(\beta^{t+1} - \theta^{t+1})$ in iteration $t + 1$. See how this controls the shrinkage towards equal solutions β and θ.
The β problem: take derivatives wrt β:

$$-X'(y - X\beta) + u + \rho(\beta - \theta) = 0$$

And therefore $\beta^{t+1} = (X'X + \rho I)^{-1}(X'y + \rho\theta^t - u^t)$ (which looks like a ridge solution update).
The θ problem: take derivatives wrt θ:

$$\lambda\nu - u^t - \rho(\beta - \theta) = 0$$

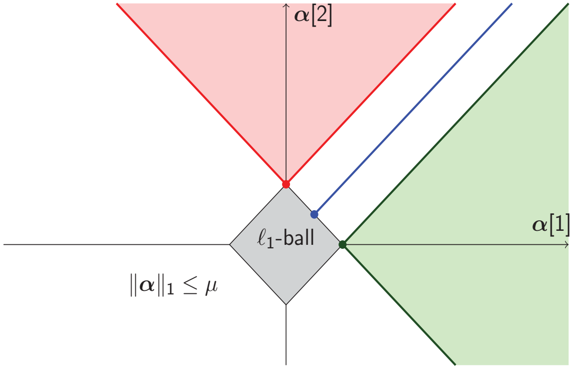
which gives us $\theta^{t+1} = ST(\beta^t + u^t/\rho, \lambda/\rho)$

L1-penalized regression

So now you've seen that the L1-penalty induces model selection or *sparsity*. The reason is the non-smooth shape of the penalty region. Here follows some illustration from Julien Mairal's excellent slides.

Why does the ℓ_1-norm induce sparsity?

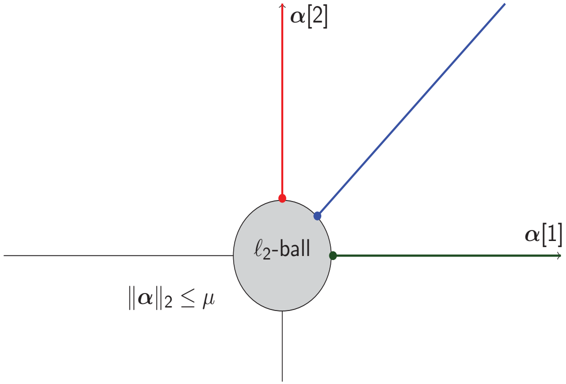
Regularizing with the ℓ_1-norm



The projection onto a convex set is “biased” towards singularities.

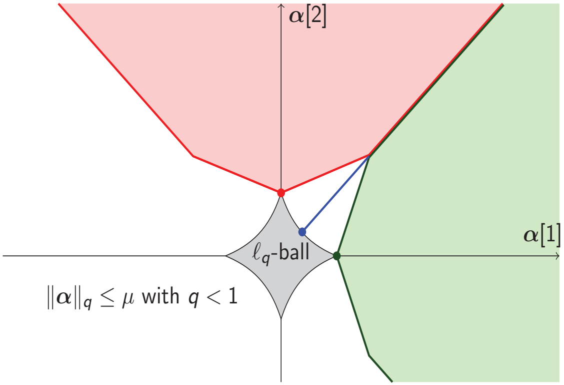
Why does the ℓ_1-norm induce sparsity?

Regularizing with the ℓ_2-norm



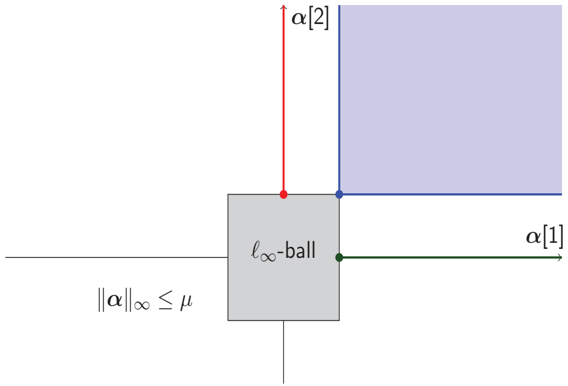
The ℓ_2-norm is isotropic.

Non-convex sparsity-inducing penalties



Why does the ℓ_1-norm induce sparsity?

Regularizing with the ℓ_∞-norm



The ℓ_∞-norm encourages $|\boldsymbol{\alpha}[1]| = |\boldsymbol{\alpha}[2]|$.

Penalized regression

The point is that we can be a bit more adventurous about constructing penalty regions in order to generate a desired type of sparsity.

Group-penalized regression

Let's say there's a natural grouping of the variables: e.g. factor levels of a categorical level, source, etc. We want to include a group of variables and not select them separately.
We can achieve this by using a *group penalty* instead. Consider the case of K groups of variables:

$$\min_{\beta} \frac{1}{2}||y - X\beta||^2 + \lambda \sum_{k=1}^{K} ||\beta_k||_2$$

where $||\beta_k||_2 = \sqrt{\sum_{j \in k} |\beta_j|^2}$, i.e. we penalize the average β value within each group.

Group-penalized regression

What's the effect of this penalty? We use coordinate descent at the group level to find out. Take derivatives wrt to group k

$$-X_k'(y - \sum_{j \ not \in k} X_j\beta_j - X_k\beta_k) + \lambda\nu_k = -X_k'(r_k - X_k\beta_k) + \lambda\nu_k$$

where r_k is the partial residual where we hold all β not in group k fixed and

$$\nu_k = \begin{cases} \frac{\beta_k}{||\beta_k||_2} & if \quad ||\beta_k||_2 \neq 0 \\ \{\nu : ||\nu||_2 \leq 1\} & if \quad ||\beta_k||_2 = 0 \end{cases} \tag{4}$$

Group-penalized regression

If $||\beta_k||_2 = 0$ we get $-X_k' r_k + \lambda\nu = 0$
and so the condition for $||\beta_k||_2 = 0$ is $||X_k r_k||_2 \leq \lambda$
If $||\beta_k||_2 \neq 0$ we get $-X_k r_k + X_k' X_k \beta_k + \lambda \frac{\beta_k}{||\beta_k||_2}$
which starts to take on the familiar form...
and so the solution is

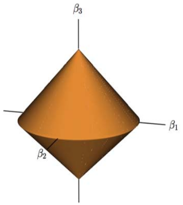
$$\beta_k = (X_k' X_k + \frac{\lambda}{||\beta_k||_2} I)^{-1} X_k r_k$$

if $||X_k r_k|| > \lambda$ and 0 otherwise.
Notice, this is like a soft threshold of a ridge estimate. The penalty for group k depends on the size of the previous iteration estimates. So this induces a group-sparsity. Why?

Group-penalized regression

Group LASSO Ball

- If $D_k \in \mathbb{R}^{N\times 1}$, then $||X||_{2,1} = ||X||_1$
- That is, if all the groups are singletons, the optimization problem reduces to LASSO.
- Group LASSO ball shares attributes of both ℓ_2 and ℓ_1 balls.



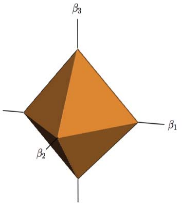


Figure 2: Group LASSO LASSO

From a talk by Hojjat Akhondi-Asl, UCL.

Elastic net

Lasso struggles when covariates are correlated and tends to pick only one of them even if both are related to the outcome. We can form groups of correlated variables and run group-lasso (Howard Bondell and others) or we can let the data decide for us and "helping" a bit by altering the penalty as follows:

$$\min_{\beta} \frac{1}{2}||y - X\beta||^2 + (1-\alpha)\lambda\frac{1}{2}||\beta||_2^2 + \alpha\lambda||\beta||_1$$

As you can see, this uses both an $L1$ and an $L2$ penalty on β. This penalty strategy is called the *elastic net*.

Elastic net

We go through the machinery again and arrive at a soft-thresholding solution

$$\beta_j = \frac{ST(X_j' r_j, \lambda\alpha)}{X_j' X_j + \lambda(1-\alpha)}$$

What tends to happen is that the bigger you make the $L2$ penalty (small α) the more elastic net with add groups of variables together into the model (see class demo).

Elastic net

$$\frac{1}{2}||y - X\beta||^2 + (1-\alpha)\lambda||\beta||_2^2 + \alpha\lambda||\beta||_1$$

- Zou and Hastie proposed the elastic net in 2005
- The motivation was the observation that if variables are correlated, lasso tends to include only one variable from the group
- Furthermore, it was seen that lasso performs poorly compared to ridge regression when there are correlated predictors present.
- Why not group lasso? Sometimes it is not so easy to determine the groups - another tuning process where we cluster variables first.
- How much grouping depends on the pairwise correlations between variables in comparison to the penalty $\lambda(1-\alpha)$.

Group-penalized regression

We return for a moment to the group-penalty. What if we don't want the whole group but only "encourage" it but use fewer variables if possible?
We impose an additional sparsity constraint on the individual parameters too!
Consider the case of K groups of variables:

$$\min_{\beta} \frac{1}{2}||y - X\beta||^2 + \lambda(1-\alpha)\sum_{k=1}^{K}||\beta_k||_2 + \lambda\alpha||\beta||_1$$

This looks a bit like elastic net actually!

Group-penalized regression

What's the effect of this penalty? We use coordinate descent at the group level to find out. Take derivatives wrt to group k

$$-X_k'(r_k - X_k\beta_k) + (1-\alpha)\lambda\nu + \alpha\eta$$

where r_k is the partial residual where we hold all β not in group k fixed and

$$\nu = \begin{cases} \frac{\beta_k}{||\beta_k||_2} & \textit{if} \quad ||\beta_k||_2 \neq 0 \\ \{\nu : ||\nu||_2 \leq 1\} & \textit{if} \quad ||\beta_k||_2 = 0 \end{cases} \tag{5}$$

$$\eta_j = \begin{cases} \textit{sign}(\beta_j) & \textit{if} \quad \beta_j \neq 0 \\ \{\eta : |\eta| \leq 1\} & \textit{if} \quad \beta_j = 0 \end{cases} \tag{6}$$

Group-penalized regression

If $||\beta_k||_2 = 0$ we get $-X_k' r_k + \lambda(1-\alpha)\nu + \alpha\lambda\eta = 0$
and so the condition for $||\beta_k||_2 = 0$ is $ST(X_k r_k, \alpha\lambda) \leq (1-\alpha)\lambda$
If $||\beta_k||_2 = 0$ and $\beta_j = 0$ we get the condition $||X_j r_j|| \leq \alpha\lambda$.
If $||\beta_k||_2 \neq 0$ and $\beta_j \neq 0$ we get

$$-X_j r_j + X_j' X_j \beta_j + \lambda(1-\alpha)\frac{\beta_k}{||\beta_k||_2} + \lambda\alpha\eta$$

which starts to take on the familiar form...
and so the solution is

$$\beta_j = \frac{ST(X_j' r_j, \lambda\alpha)}{X_j' X_j + (1-\alpha)\lambda/||\beta_k||_2}$$

Summary

- Structured penalty to induce desired sparsity
- Simple coordinate descent works for big problems but perhaps not the most efficient
- ADMM is good for complex penalties since one can decompose the problem in layers
- LARS: least angle regression and resolution paths. Deriving solutions for all values of λ in one go. Idea: forward stepwise regression such that add the variable *most correlated with current residuals* until another variable takes on that role.

OSCAR

$$\frac{1}{2}||y - X\beta||^2 + \lambda||\beta||_2 + \lambda c \sum_{j<k} \max(|\beta_j|, |\beta_k|)$$

- The OSCAR penalty was proposed by Bondell and Reich in 2006.
- It tries to automate group detection just like elastic net
- but the penalty region has "corners" so also tries to make coefficients equal if they are correlated.
- The `lqa()` package has both elastic net and OSCAR implemented.

Generalized linear models

When we have categorical outcome data we can still use penalized fitting strategies.
The objective function will now be the negative log-likelihood of the data plus the penalty.
For binomial data, our model is

$$P(y = 1|X = x) = \frac{e^{x\beta}}{1 + e^{x\beta}}$$

and so we minimize

$$\sum_i (y_i(x\beta) - log(1 + e^{x\beta})) + J(\beta)$$

- This is nonlinear in β
- Use a quadratic approximation of the log-likelihood and coordinate descent
- Then the problem looks like a penalized least squares problem
- Solve this and iterate
- package `glmnet()`

Generalized linear models

What if you have more than two classes? Multinomial model.

$$P(y = k|X = x) = \frac{e^{x\beta_k}}{\sum_{j=1}^{K} e^{x\beta_j}}$$

- Notice we actually have set of coefficients $\beta_k = \{\beta_{0k}, \beta_{1k}, \cdots, \beta_{jk}\}$, one p-vector for each class.
- This $p \times K$ matrix of coefficients can be treated as separate problems
- OR if you want to have a model that is easier to interpret you let each $\beta_{jk}, k = 1, \cdots, K$ constitute a *group* so that variable j is used to predict all classes or not used at all.
- package `glmnet()`

Adaptive lasso

One problem with lasso is that while it does perform model selection it also induces bias on the coefficients. This can lead to reduced predictive performance.

- It would be better if we penalized large coefficients less and small coefficients more
- This would give as a better predictive model while still being sparse
- What we want is a so-called *oracle procedure*: one that identifies the right model and has an optimal rate of convergence ($\sqrt{n}$).
- Lasso does not have these properties

Adaptive lasso

What we need is a procedure that is nearly unbiased *and* sparse

- Fan and Li (2002) found that for this to be true we need a penalty that is 0 for large coefficients
- and singular at 0 to give us sparse estimates

Adaptive lasso

$$||y - X\beta||^2 + \lambda \sum_{j=1}^{p} w_j |\beta_j|$$

- the adaptive lasso uses weights $w_j \geq 0$ to penalize the coefficients differently
- the weights are chosen data-dependently
- We need some good initial estimates of βs to obtain good weights.

Adaptive lasso

$$||y - X\beta||^2 + \lambda \sum_{j=1}^{p} w_j |\beta_j|$$

- For oracle properties to result, the weights used are

$$w_j = \frac{1}{|\tilde{\beta}_j|^\gamma}$$

- where the initial estimates $\tilde{\beta}$ should be $\sqrt{n}$-consistent
- We could use β^{LS} if we can compute these, otherwise β^{ridge} is a popular choice

Adaptive lasso

If we use a $\sqrt{n}$-consistent estimate $\tilde{\beta}$ then one can show that if we use lasso-penalty λ_n such that $\lambda_n/\sqrt{n} \to 0$ and $\lambda_n n^{(\gamma-1)/2} \to \infty$ then

- We have consistency of variable selection
- and optimal convergence rates for the non-zero coefficients
- adaptive lasso is an oracle procedure

Adaptive lasso

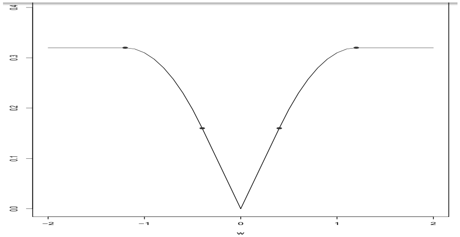
We don't even have to write a new procedure for this...

- Define $\tilde{x}_j = x_j / w_j$ for all $j = 1, \cdots, p$
- Run lasso with this new set of variables
- Output the adaptive lasso estimates as $\hat{\beta}_j^{lasso} / w_j$

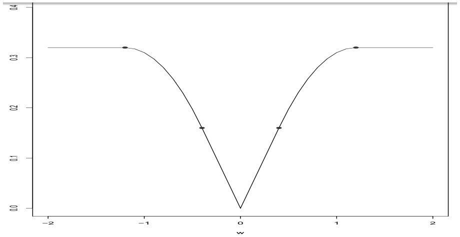
SCAD

An alternative to the adaptive lasso is the SCAD penalty (Smoothly clipped absolute deviation).

$$pen\beta, \lambda = \begin{cases} \lambda|\beta| & \textit{if} \quad |\beta| \leq \lambda \\ -\frac{(|\beta|^2 - 2a\lambda|\beta| + \lambda^2)}{2(a-1)} & \textit{if} \quad \lambda < |\beta| \leq a\lambda \\ \frac{(a+1)\lambda^2}{2} & \textit{if} \quad |\beta| > a\lambda \end{cases} \tag{7}$$



SCAD



- Notice, this penalty really looks like the ideal case!
- Singular at 0 and essentially no bias for large coefficients!
- and indeed it does have oracle properties
- BUT it's not a convex penalty so computation is harder
- use local approximation (linear or quadratic) - `lqa()` package

High-dimensional inference

- L1-penalized modeling has become enormously popular for high-dimensional problems
- We get model selection, fairly good predictions and as saw above, good point estimates
- But when we do low-dimensional modeling we usually don't feel very satisfied with just point estimates
- We want confidence intervals and p-values!

High-dimensional inference

- What are the obstacles for obtaining p-values and confidence intervals?
- Highly non-standard setting when $p > n$
- the distribution of lasso-solutions, by construction, has a point-mass at 0 and this makes bootstrapping to get standard error estimates difficult

Sample-splitting

Wasserman and Roeder (2009) proposed the following approach to obtain p-values

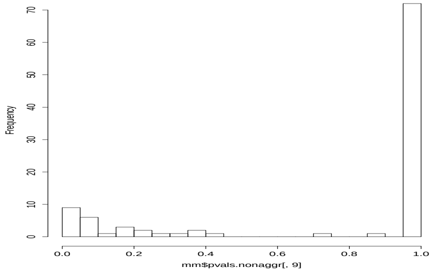
- Split the data in two sets
- Use set 1 to perform modelselection via e.g. lasso
- Use set 2 to evaluate p-values for the non-zero coefficients. This is done by running LS using only the selected variables in the model.
- For the variables not selected in set 1, set p-value to 1.

The p-values are valid because we didn't reuse the same data for selection and p-value computation.
Moreover, if we want to compute adjusted p-values that take into account multiple testing we only have to correct by the selected set of variables, not all p.

Sample-splitting

Drawback with the procedure

- Sensitive to the split so the pvalues are not reproducible
- "p-value lottery"
- Different splits leads to widely different p-values!



Multi sample-splitting

To overcome the p-value lottery we perform several random splits of data (Meinhausen et al, 2009)

- Repeat B times: split data into set 1 and set 2
- Use set 1 for selection of variables
- Use set 2 to compute p-values
- Aggregate the p-values

Hm? How to we combine B p-values (like those from the histogram above) to one final p-value?

Multi sample-splitting

The p-value estimates are not independent because the data splits overlap.

- We can use the median p-value
- Or any other quantile
- Search for the best quantile
- Implemented in package `hdi()`

De-sparsified lasso

There's been a lot of work in the last 2-3 years on the p-value and confidence interval problem of sparse estimators.
Zhang and Zhang (2014) proposed the de-sparsified lasso to come up with p-values in a high-dimensional setting.

- We start with the $p < n$ setting
- We are interested in the j-th coefficient estimate
- It turns out we can obtain the LS estimate as follows

$$\hat{\beta}_j^{LS} = \frac{y' Z_j}{X_j' Z_j}$$

where Z_j is the residual if you run a regression of X_j on all the other Xs!

De-sparsified lasso

Write out the true model

$$y = \sum_{j=1}^{J} X_j \beta_j^* + \eta$$

where β^* are the true coefficient values

- If we plug this into the estimate $\hat{\beta}_j^{LS} = \frac{y' Z_j}{X_j' Z_j}$ we see

$$\frac{y' Z_j}{X_j' Z_j} = \beta_j^* + \sum_{k \neq j} \beta_k^* \frac{X_k' Z_j}{X_j' Z_j} + \frac{\eta' Z_j}{X_j' Z_j}$$

- When we have run regression with LS the residuals Z_j are orthogonal to the other variables X_k and so we see that terms 2 on the right hand side is 0.
- What happens when $p > n$?
- Then this doesn't work since residuals Z_j are 0

De-sparsified lasso

Idea (Zhang and Zhang, 2014): Use a *regularized regression* of X_j on the other Xs!

- If we plug this into the estimate $\hat{\beta}_j^{LS} = \frac{y'Z_j}{X_j'Z_j}$ we see

$$\frac{y'Z_j}{X_j'Z_j} = \beta_j^* + \sum_{k \neq j} \beta_k^* \frac{X_k'Z_j}{X_j'Z_j} + \frac{\eta' Z_j}{X_j'Z_j}$$

- Now term 2 does not go away and therefore we now have a biased estimate of β_j^*
- Bias correction

$$\hat{\beta}_j = \frac{y'Z_j}{X_j'Z_j} - \sum_{k \neq j} \hat{\beta}_k \frac{X_k'Z_j}{X_j'Z_j}$$

where we use the lasso-estimates $\hat{\beta}_k$

De-sparsified lasso

Zhang and Zhang (2014) and van de Geer at al (2014) has derived the distribution for the bias-corrected estimate as

$$\sqrt{n}(\hat{\beta} - \beta^*) \sim N_p(0, W)$$

- Since from above we have

  $$\sqrt{n}(\hat{\beta}_j - \beta^*_k) = \frac{\sqrt{n}\eta' Z_j}{n^{-1} X_j' Z_j} + R$$

  where R can be shown to be neglible under sparsity assumptions on β^* and structure on X
- Then we can derive the distribution variance W from the term involving η as

  $$W_{jk} = \sigma_\eta \frac{Z_j' Z_k}{(X_j' Z_j)(X_k' Z_k)}$$
- And now we can compute p-values for every β!!!

BIAS-CORRECTED RIDGE REGRESSION

Another proposal by Buhlmann (2013) uses a bias-corrected ridge estimate

- Here we start with the ridge regression estimate
- Then we perform bias-correction using lasso-estimates
- Buhlmann (2013) derive the sampling distribution for the bias-corrected estimates
- And now we can compute p-values for every β!!!
- Computationally cheaper than the de-sparsified lasso
- Tuning parameters need to selected - CV can be used or other criteria (see journal paper)
- package `hdi()`

Correlated variables

In practice, we often have highly correlated variables in our data sets. This was the motivation for group selection in elastic net or group lasso.
When we have correlated variables this translates to higher estimation variance within the group, wider confidence intervals and lower power of detection.

- Group testing is one solution
- We can group the variables together based on their pairwise correlations, e.g. via hierarchical clustering
- We can then compute p-values for each group.
- How do we we generate group-p-values?
- In de-sparsified lasso and ridge we adjust the individual p-values by the number of tests performed (p) and the then use the minimum adjusted p-value within the group for group decisions.

Correlated variables

Meinhausen (2013) proposes a multi-split testing of groups as follows.

- We use multi-sample splitting to construct confidence intervals for the l1-norm of a group.
- If the lower bound of this confidence interval is larger than 0, we reject the null-hypothesis for this group.
- `hdi()` package illustrates the group tests with a hierarchical tree (see demo)

Summary

- Lasso can be made to perform very well with adjustments: elastic net (and the like) for correlated variables and adaptive lasso to remove bias.
- Can apply adaptive lasso to elastic net and for generalized linear models also.
- `lqa()` package has all of these procedures, and more, implemented.
- New research on high-dimensional inference: de-sparsified lasso, bias-corrected ridge can be used to test individual variable contributions or assess groups of variables in high-dimensional settings
- Multi-sample splitting another alternative (easier to generalize to other penalties).
- `hdi()` package has all of these procedures implemented.