AME

Week 5: High Dimensional Models I

Sophie Bindslev, October 2022

UNIVERSITY OF COPENHAGEN

Today's Plan

- High Dimensional World
- OLS
- Lasso
- Standardization
- Tuning: choice of penalty
- Your time to shine!

High Dimensional Models

- Linear model

$$\mathbf{Y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\epsilon} \tag{1}$$

  where $\mathbf{Y}$ is $N \times 1$, $\mathbf{X}$ is $N \times p$ and $\boldsymbol{\beta}$ is $p \times 1$

- Usually we consider scenarios where $N >> p$ and where asymptotics are derived for $N \to \infty$ such that $\frac{p}{N} \to 0$
- In high dimensional settings $\frac{p}{N}$ is non-negligible

OLS poor prediction

- Suppose our main aim is to predict $\mathbf{Y}$ given a set of covariates $\mathbf{X}$
- If $\frac{p}{N}$ is non-negligble OLS will perform poorly as the (out of sample) prediction error is

$$E(\frac{1}{N}\sum_{i=1}^{N}(\mathbf{X}_i\hat{\beta}-\mathbf{X}_i\beta)^2)=\frac{\sigma^2 p}{N} \tag{2}$$

  where σ^2 is the variance of the IID error term ϵ
- NB we have moved to a machine learning world where prediction rather than causality is the primary aim
- This assumes that OLS is defined. For $p > N$ the rank condition fails since $rank(\mathbf{X}'\mathbf{X}) = N < p$

Lasso

- Rescue comes from believing sparsity applies i.e. only a subset $J < p$ of β are non-zero
- The Lasso estimator performs variable selection and regularisation

$$\hat{\beta}(\lambda) = \underset{b \in \mathbb{R}^p}{argmin} \frac{1}{N} \sum_{i=1}^{N} (Y_i - \mathbf{X}_i b)^2 + \lambda \mid\mid b \mid\mid_1 \tag{3}$$

$$where \quad \mid\mid b \mid\mid_1 = \sum_{j=1}^{p} \mid b_j \mid \tag{4}$$

- You can implement the Lasso for given penalty levels λ using the `sklearn.linear_model.Lasso` package (where α is the penalty level and is divided by 2 - remember to divide your λs by 2!)

Standardize Data

- Lasso is sensitive to the scaling of variables. Variables with bigger standard deviations, say, because they are measured in 1DKK rather than 1,000DKK will be penalised more
- To avoid this unintended effect on variable selection performed by Lasso we standardize the regressors
- We bring them all on the same scale by

$$\tilde{\mathbf{X}} = \frac{\mathbf{X} - \bar{\mathbf{X}}}{\boldsymbol{\sigma}_X} \tag{5}$$

  where $\bar{\mathbf{X}}$ and $\boldsymbol{\sigma}_X$ are the mean and standard deviation of $\mathbf{X}$, respectively
- What does this imply for the interpretation of β?

Tuning: Penalty Selection

- We need to pick a penalty level λ which will be key for variable selection performed by Lasso
- Methods for selecting λ we will consider include
  - Cross Validation (CV)
  - Bickel-Ritov-Tsybakov Rule (BRT)
  - Belloni-Chen-Chernozhukov-Hansen Rule (BCCH)

Cross Validation

- Divides sample into K subsamples of equal size
- For each subsample $k = 1, \dots K$
  - CV uses subsample k for validation and the remaining subsamples for training
  - Computes (mis)fit $F_k(\lambda) = \frac{1}{N-M}\sum_{i=M+1}^{N}(Y_i - \mathbf{X}_i\hat{\beta}(\lambda))^2$ where M is the number of observations in each subsample k
- The CV penalty level is then

$$\hat{\lambda}^{CV} = \underset{\lambda}{argmin} \sum_{k=1}^{K} F_k(\lambda) \tag{6}$$

- Implementation: `sklearn.linear_models.LassoCV(cv = K)` where K is the number of folds

Bickel-Ritov-Tsybakov Rule (BRT)

- BRT relies on two conditions:
  A) ϵ is independent of $\mathbf{X}$ and homoskedastic
  B) variance σ^2 of ϵ is known
- To compute $\hat{\lambda}^{BRT}$ we
  1) choose $\alpha \in (0, 1)$, usually $\alpha = 0.05$ the prob of being inside/outside the error term
  2) choose $c > 1$, typically $c = 1.1$ defines the upper bound of the error how high an error you are willing to accept
  3) use

$$\hat{\lambda}^{BRT} = \frac{2c\sigma}{\sqrt{N}}\Phi^{-1}(1 - \frac{\alpha}{2p})\sqrt{\max_{1\leq j\leq p}\frac{1}{N}\sum_{i=1}^{N}\mathbf{X}_{i}^{2}} \qquad (7)$$

everything has variance 1 and drops out

where Φ is the standard normal CDF

- What happens to this formula when we standardize our $\mathbf{X}$?

Belloni-Chen-Chernozhukov-Hansen Rule (BCCH)

- BCCH allows heteroskedasticity and requires no preliminary knowledge of the variance of the error terms
- To compute $\hat{\lambda}^{BCCH}$ we
  1) choose $\alpha \in (0,1)$ and c as for BRT
  2) obtain pilot Lasso $\hat{\beta}(\hat{\lambda}^{pilot}) = \hat{\beta}^{pilot}$ where

$$\hat{\lambda}^{pilot} = \frac{2c}{\sqrt{N}} \Phi^{-1}(1 - \frac{\alpha}{2p}) \sqrt{\max_{1 \leq j \leq p} \frac{1}{N} \sum_{i=1}^{N} (Y_i - \bar{Y})^2 \mathbf{X}_i^2} \qquad (8)$$

  3) obtain residuals $\hat{\epsilon}_i = Y_i - \mathbf{X}_i \hat{\beta}^{pilot}$ from pilot-Lasso and compute penalty:

sigma^2 is gone

$$\hat{\lambda}^{BCCH} = \frac{2c}{\sqrt{N}} \Phi^{-1}(1 - \frac{\alpha}{2p}) \sqrt{\max_{1 \leq j \leq p} \frac{1}{N} \sum_{i=1}^{N} \hat{\epsilon}_i \mathbf{X}_i^2} \qquad (9)$$

Your time to shine!

- Solve the problem set
- Tip #1: take a look at the documentations for these functions: `sklearn.linear_model.Lasso`, `sklearn.linear_model.LassoCV` and `sklearn.preprocessing.PolynomialFeatures`
- Tip #2: these functions are also implemented in Jesper's slides so if you need more info you can look there
- Features such as `.predict_`, `.alpha_` and `.coef_` are especially useful when e.g. computing residuals