05 - Regression and Regularization

ml4econ, HUJI 2021

Itamar Caspi

April 11, 2021 (updated: 2021-04-12)

Packages and setup

Use the {pacman} package that automatically loads and installs packages if necessary:

```r
if (!require("pacman")) install.packages("pacman")

pacman::p_load(
  tidyverse,   # for data wrangling and visualization
  knitr,       # for displaying nice tables
  broom,       # for tidying estimation output
  here,        # for referencing folders and files
  glmnet,      # for estimating lasso and ridge
  gamlr,       # for forward stepwise selection
  pls,         # for estimating PCR and PLS
  elasticnet,  # for estimating PCR and PLS
  ggfortify
)
```

Set a theme for `ggplot` (Relevant only for the presentation)

```r
theme_set(theme_grey(20))
```

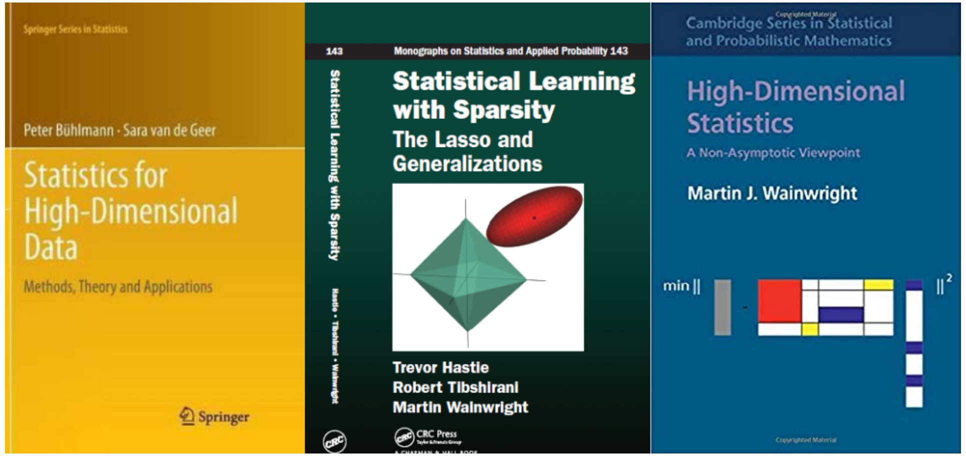
And set a seed for replication

Resources on hige-dimensional statsitics

- *Statistical Learning with Sparsity - The Lasso and Generalizations* (Hastie, Tibshirani, and Wainwright), (PDF available online)

- *Statistics for High-Dimensional Data - Methods, Theory and Applications* (Buhlmann and van de Geer)

- *High Dimensional Statistics - A Non-Asymptotic Viewpoint* (Wainwright)



Springer Series in Statistics

Peter Bühlmann · Sara van de Geer

Statistics for High-Dimensional Data

Methods, Theory and Applications

Springer

143 Monographs on Statistics and Applied Probability 143

Statistical Learning with Sparsity

The Lasso and Generalizations

Trevor Hastie
Robert Tibshirani
Martin Wainwright

CRC Press

Statistical Learning with Sparsity

Hastie • Tibshirani • Wainwright

Cambridge Series in Statistical and Probabilistic Mathematics

High-Dimensional Statistics

A Non-Asymptotic Viewpoint

Martin J. Wainwright

Outline

- Linear regression
- Penalized regression
- Subset selection
- Shrinkage
- Dimension Reduction

Linear Regression

Econometrics

In econometrics, we typically assume a "true" linear data generating process (DGP):

$$y_i = \beta_0 + \sum_{j=1}^{p} x_{ij}\beta_j + \varepsilon_i$$

where y_i is the outcome variable, $x_{ij}, \ldots, x_{ip}$ is a set of explanatory or control variables (+ interactions, polynomials, etc.), and ε_i is the regression error.

sample : $\{(x_1, \ldots, x_p, y_i)\}_{i=1}^{n}$

Estimation

Ordinary least squares minimizes

$$\min_{\beta_0,\beta} \sum_{i=1}^{N} \left(y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j \right)^2$$

The emphasis here is on in-sample fit (minimize residual sum of squares).

Typical workflow:

- impose identifying assumptions (causal claims).
- estimate $\beta_0, \ldots, \beta_p$ using the entire sample.
- assume a random sample from a larger population.
- hypothesis testing.

Supervised Learning

Consider the following linear data generating process (DGP):

$$y_i = \beta_0 + \sum_{j=1}^{p} x_{ij}\beta_j + \varepsilon_i$$

where y_i is the predicted (response) variable, $x_{i1}, \ldots, x_{ip}$ is a set of "features", and ε_i is the irreducible error.

- **Training set** : $\{(x_{1i}, \ldots, x_{ip}, y_i)\}_{i=1}^{n}$
- **Test set** : $\{(x_{1i}, \ldots, x_{ip}, y_i)\}_{i=n+1}^{m}$

> Typical assumptions: (1) independent observations; (2) stable DGP across training *and* test sets.

Our object of interest is $\hat{y}_i$, predicting unseen data.

Dffierent objective, different approach

To illustrate how these two approaches (estimation vs. prediction) differ, consider the following data generating process[1]:

$$y_i = \beta_0 + \beta_1 x_i + \varepsilon_i, \qquad \varepsilon_i \sim N(0, \sigma^2)$$

where $\beta_0 = 0$ and $\beta_1 = 2$.

Next, suppose you get a sample of size N and use it to estimate the model and get (standard errors in parentheses):

$$\begin{array}{rcccl} y_i = & 0.0 & + & 2.0 & \times x_i + \hat{\varepsilon}_i \\ & (0.2) & & (10.0) & \end{array}$$

Given a new unseen x^0 and the output above, how would you predict y^0?

[1] Adapted from Susan Athey's lecture.

Standard errors and prediction accuracy

- OLS is lexicographic in the sense that it first ensure unbiasedness, then efficiency.
- OLS is unbiased, hence you would be right *on average*, but. Is that what we want?
- If your estimated coefficient is noisy (high std. error), so will be your prediction (high variability).
- Bias-variance trade off again.
- Prediction is about balancing bias and variance.

NOTE: in multivariate regression, things are more complicated due to the correlation structure of the x's.

Illustration: Browse data

In this lecture, we will use Matt Taddy's **browser dataset** (available in our repo) which contains web browsing logs for 10,000 people. Taddy has extracted a year's worth of browser logs for the 1000 most heavily trafficked websites. Each browser in the sample spent at least $1 online in the same year.

The goal of estimation: predict spending from browser history:

$$\log(spend_i) = \beta_0 + \beta' \boldsymbol{visits}_i + \varepsilon_i, \qquad \text{for } i = 1, \ldots, n$$

where $\boldsymbol{visits}_i$ is a vector site-visit percentage. This model can be used to segment expected user budget as a function of browser history.

Load data

In this lecture we will only use a sample from the browser dataset: 250 websites and 1000 users.

```r
browser <- here("05-regression-regularization/data","browser-all.csv") %>%
  read_csv()
```

Log spending by the first user and the fraction of times she spent on the first 4 websites:

```r
browser[6, 1:5]
```

```
## # A tibble: 1 x 5
##   log_spend `123greetings.com` `204.95.60.12` `2o7.net` `65.115.67.11`
##       <dbl>              <dbl>          <dbl>     <dbl>          <dbl>
## 1      7.74                  0         0.0483    0.0483         0.0322
```

NOTE: The design matrix in this case is sparse.

Data to matrices

We will soon see that it is useful to transform the data to response and features vector and matrix, respectively:

```r
browser_mat <- browser %>%
  as.matrix()

Y_browser <- browser_mat[, 1]     # response
X_browser <- browser_mat[, 2:201] # features
```

OLS results

Estimate the model using `lm()`

```
lm_fit <- lm(log_spend ~ ., data = browser)
```

Show estimation output, sorted by p-values

```
lm_fit %>%
  tidy() %>%
  arrange(p.value) %>%
  head(3) %>%
  kable(format = "html", digits = 2)
```

term	estimate	std.error	statistic	p.value
$bizrate.com - o01$	1.86	0.25	7.54	0
staples.com	1.33	0.22	5.91	0
victoriassecret.com	1.45	0.25	5.91	0

Model perfoemance

What is the training-set MSE?

```
lm_fit %>%
  augment() %>%
  summarise(mse = mean((log_spend - .fitted)^2)) %>%
  kable(format = "html", digits = 3)
```

mse
2.075

This is clearly an *underestimate* of the test set MSE.

Penalized linear regression

Estimation

Penalized (or *regularized*) sum of squares solves

$$\min_{\beta_0,\beta} \sum_{i=1}^{N} \left(y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j \right)^2 \text{ subject to } R(\beta) \leq t$$

where $R(\cdot)$ is a penalty function that measures the **expressiveness** of the model.

As the number of features grows, linear models become *more* expressive.

Notation: Norms

Suppose $\boldsymbol{\beta}$ is a $\boldsymbol{p} \times \mathbf{1}$ vector with typical element β_i.

- The ℓ_0-norm is defined as $||\boldsymbol{\beta}||_0 = \sum_{j=1}^{p} \mathbf{1}_{\{\beta_j \neq 0\}}$, i.e., the number of non-zero elements in $\boldsymbol{\beta}$.
- The ℓ_1-norm is defined as $||\boldsymbol{\beta}||_1 = \sum_{j=1}^{p} |\beta_j|$.
- The ℓ_2-norm is defined as $||\boldsymbol{\beta}||_2 = \left(\sum_{j=1}^{p} |\beta_j|^2\right)^{\frac{1}{2}}$, i.e., Euclidean norm.

Commonly used penaltiy functions

It is often convenient to rewrite the regularization problem in the Lagrangian form:

$$\min_{\beta_0, \beta} \sum_{i=1}^{N} \left(y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j \right)^2 + \lambda R(\beta)$$

NOTE: There is one-to-one correspondence between λ and t.

Method	$R(\boldsymbol{\beta})$
OLS	0
Subset selection	$\|\|\boldsymbol{\beta}\|\|_0$
Lasso	$\|\|\boldsymbol{\beta}\|\|_1$
Ridge	$\|\|\boldsymbol{\beta}\|\|_2^2$
Elastic Net*	$\alpha\|\|\boldsymbol{\beta}\|\|_1 + (1-\alpha)\|\|\boldsymbol{\beta}\|\|_2^2$

[*] Will not be covered in this lecture. Essentially, a fancy name for combining ridge and lasso.

Best subset selection

Our goal

$$\min_{\beta_0,\beta} \sum_{i=1}^{N} \left(y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j \right)^2 \text{ subject to } \|\boldsymbol{\beta}\|_0 \leq t$$

In words: select the best model according to some statistical criteria, among all possible combination of t feature or less.

Best subset selection algorithm

1. For $k = 0, 1, \ldots, p$

   1.1 Fit all models that contain exactly k predictors. If $k = 0$, the forecast is the unconditional mean.

   1.2 Pick the best (e.g, highest R^2) among these models, and denote it by $\mathcal{M}_k$.

2. Optimize over $\{\mathcal{M}_0, \ldots, \mathcal{M}_p\}$ using cross-validation (or other criteria)

Issues:

1. The algorithm is very slow: at each step we deal with $\binom{p}{k}$ models ("N-P complete".)
2. The prediction is highly unstable: the subsets of variables in $\mathcal{M}_{10}$ and $\mathcal{M}_{11}$ can be very different from each other, leading to high variance (the best subset of $\mathcal{M}_3$ need not include any of the variables in best subset of $\mathcal{M}_2$.)

Faster subset selection algorithms

Instead of estimating all possible combinations, follow a particular path of models:

- Forward stepwise selection: start simple and expand (feasible even if $p > n$)
- Backward stepwise selection: start with the full model and drop features (not recommended)

Forward stepwise algorithm

1. Let $\mathcal{M}_0$ denote the null model, which contains just an intercept.

2. For $k = 0, \ldots, p - 1$:

   2.1 Consider all $p - k$ models that augment the predictors in $\mathcal{M}_k$ with one additional predictor.

   2.2 Choose the best among these $p - k$ models, and call it $\mathcal{M}_{k+1}$. Here best is defined as having highest R^2

3. Select a single best model from among $\mathcal{M}_0, \ldots, \mathcal{M}_p$ using cross-validation.

> This is our first example of a **greedy algorithm**: a making the locally optimal selection at each stage with the intent of finding a global optimum.

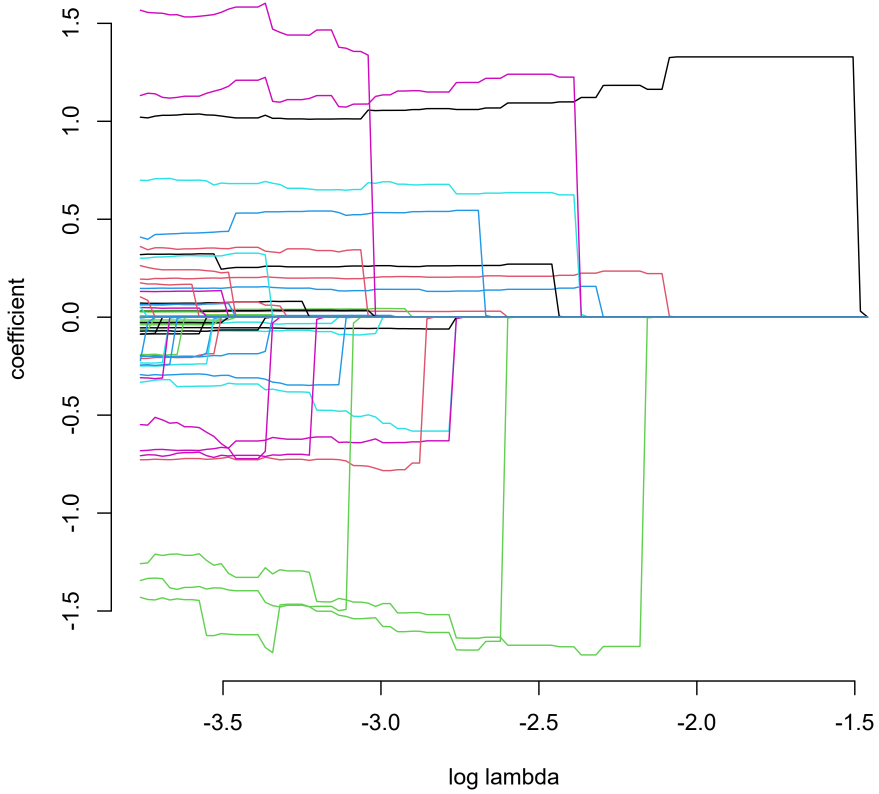
Stepwise using `gamlr`

`{gamlr}` is an R package that enables you, among other things, to estimate a forward stepwise regression model.

```
fit_step <- gamlr(X_browser, Y_browser, gamma=Inf, l
plot(fit_step, df=FALSE, select=FALSE)
```

The figure on the right shows the value of the coefficients along the forward stepwise selection path.

Notice how the jagged are the solution paths. This discontinuity is the cause for instability in subset selection algorithms.



Shrinkage

Prerequisite: centering and scaling

In what follows, we assume that each feature is centered and scaled to have mean zero and unit variance, as follows:

$$\frac{x_{ij} - \widehat{\mu}_i}{\widehat{\sigma}_i}, \qquad \text{for } j = 1, 2, \dots, p$$

where $\widehat{\mu}_i$ and $\widehat{\sigma}_i$ are the estimated mean and standard deviation of x_i estimated over the *training* set.

NOTE: This is not important when using OLS (why?)

The ridge regression

Ridge regression was introduced into the statistics literature by Hoerl and Kennard (1970).

The optimization problem:

$$\min_{\beta_0,\beta} \sum_{i=1}^{N} \left(y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j \right)^2 + \lambda \|\boldsymbol{\beta}\|_2^2$$

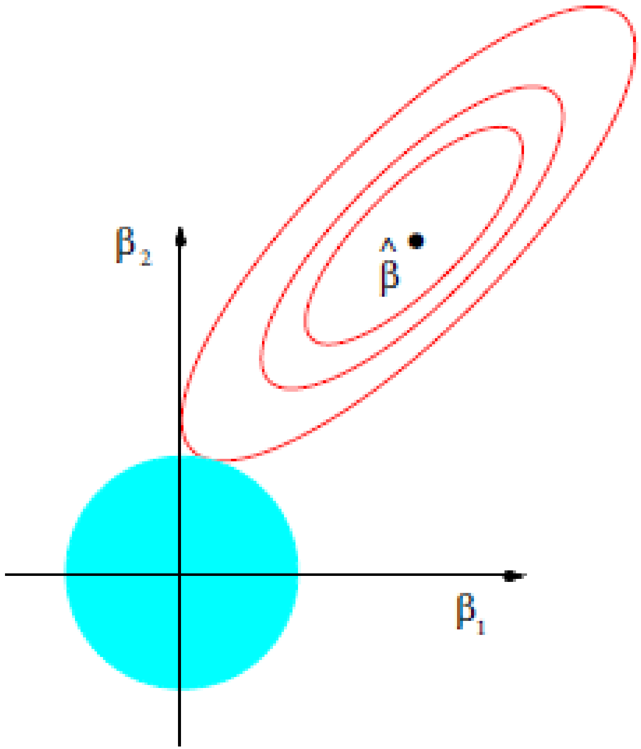
or in a budget constraint form:

$$\min_{\beta_0,\beta} \sum_{i=1}^{N} \left(y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j \right)^2 \text{ subject to } \|\boldsymbol{\beta}\|_2^2 \leq t$$

Ridge puts a "budget constraint" on the sum of squared betas. This constraint incorporate the cost of being "too far away" from the null hypothesis of $\beta_j = 0$ (what if this assumption is wrong?)

Illistration of ridge in 2-D

Contours of the error and constraint functions for ridge regression. The solid blue area is the constraint region, $\beta_1^2 + \beta_2^2 \leq t$, while the red ellipses are the contours of the RSS and $\widehat{\beta}$ is the OLS estimator.



Source: James et al. (2017)

The solution

The problem in matrix notation:

$$\min_{\boldsymbol{\beta}} \ (\mathbf{y} - \mathbf{X}\boldsymbol{\beta})'(\mathbf{y} - \mathbf{X}\boldsymbol{\beta}) + \lambda\boldsymbol{\beta}'\boldsymbol{\beta}$$

The ridge estimator is given by

$$\widehat{\boldsymbol{\beta}}^{\mathrm{R}} = \left(\mathbf{X}'\mathbf{X} + \lambda\mathbf{I}\right)^{-1}\mathbf{X}'\mathbf{y}$$

NOTE: We can have a solution even if $\mathbf{X}$ is not of full rank (e.g., due to multicollinearity) since $\mathbf{X}'\mathbf{X} + \lambda\mathbf{I}$ is non-singular (since $\lambda > 0$.)

Bayesian interpretaion of ridge

- Consider the regression

$$y_i \sim N\left(\mathbf{x}_i' \boldsymbol{\beta}, \sigma^2\right)$$

  where we assume that $\boldsymbol{\sigma}$ is known.

- Suppose we put an independent prior on each β_j

$$\beta_j \sim N\left(0, \tau^2\right)$$

- Then, the posterior mean for $\boldsymbol{\beta}$ is

$$\widehat{\boldsymbol{\beta}}_{\text{posterior}} = \left(\mathbf{X}'\mathbf{X} + \frac{\sigma^2}{\tau^2}\mathbf{1}\right)^{-1} \mathbf{X}'\mathbf{y}$$

- Hence, $\lambda = \frac{\sigma^2}{\tau^2}$.

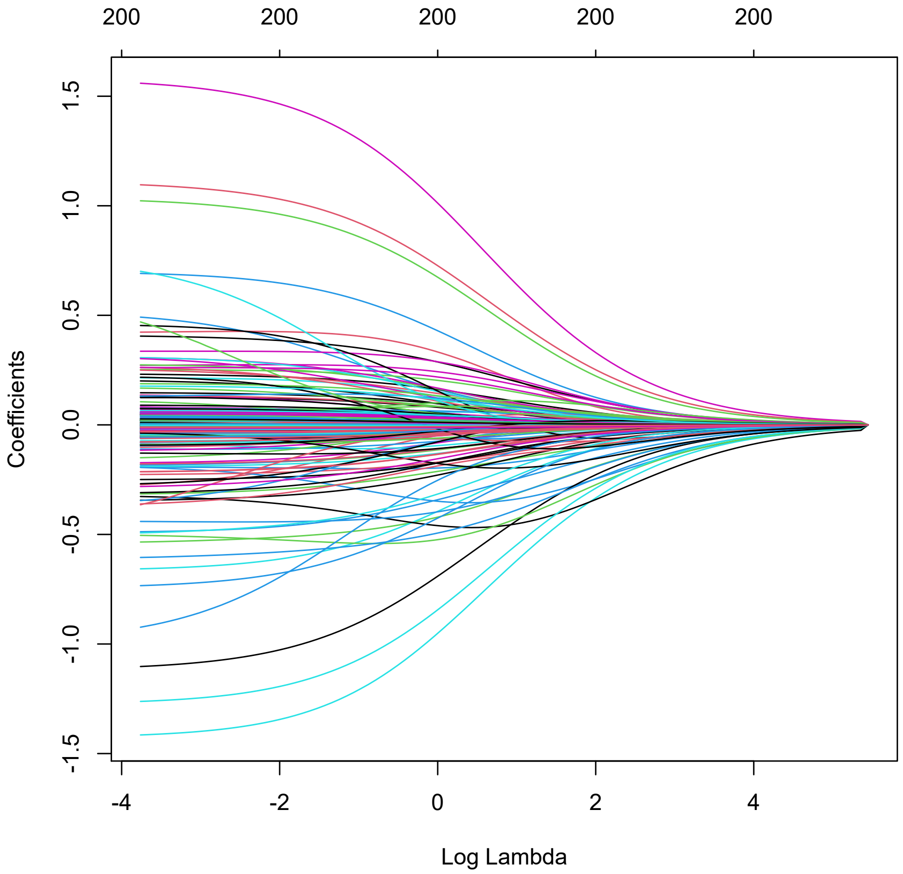
Estimating ridge using `glmnet`

The `{glmnet}` R package enables you to estimate ridge regression, along with its path for λ.

(Note that for estimating the ridge, we need to set `alpha`, the elastic net parameter to 0.)

```
fit_ridge <- glmnet(
  x = X_browser,
  y = Y_browser,
  alpha = 0
)
plot(fit_ridge, xvar = "lambda")
```

The figure on the right shows the ridge **regularization path**, i.e., the values of the



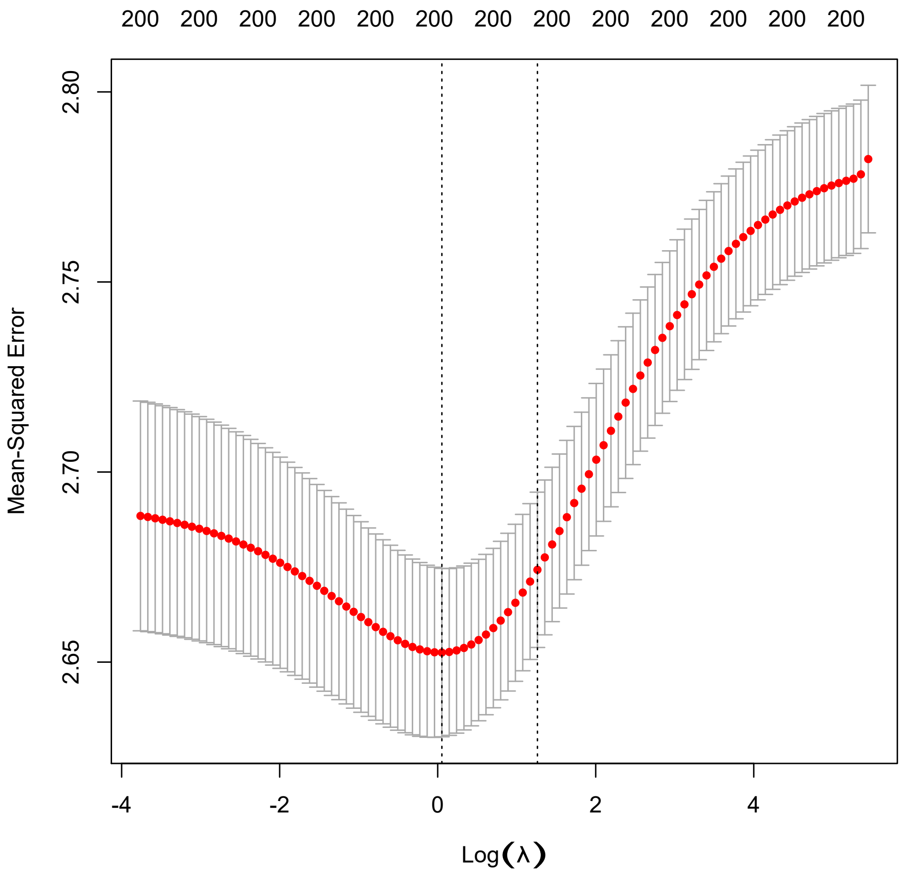
Tuning λ

The function `cv.glmnet()` automates cross validation for the ridge regression.

Note that you can change the default number of folds (10) by setting the `nfolds` argument.

```
cv_ridge <- cv.glmnet(x = X_browser, y = Y_browser,
plot(cv_ridge)
```

- *Left* dotted vertical line: λ with min MSE
- *Right* dotted vertical line: the biggest λ with MSE no more than one SE away from the minimum



The lasso

Lasso (least absolute shrinkage and selection operator) was introduced by Tibshirani (1996). The optimization problem:

$$\min_{\beta_0,\beta} \sum_{i=1}^{N} \left(y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j \right)^2 + \lambda \|\boldsymbol{\beta}\|_1$$

Lasso puts a "budget constraint" on the sum of *absolute* β's.

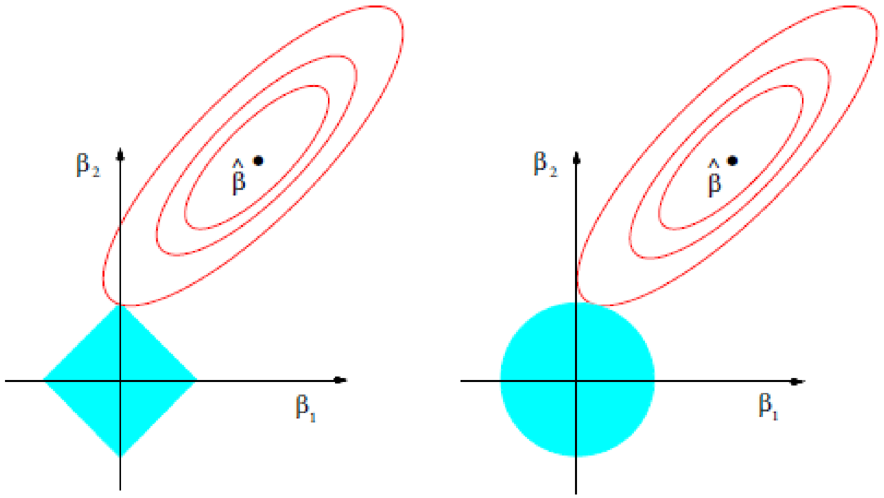
Unlike ridge, the lasso penalty is linear (moving from 1 to 2 is the same as moving from 101 to 102.)

A great advantage of the lasso is that performs model selection - it zeros out most of the β's in the model (the solution is *sparse*.)

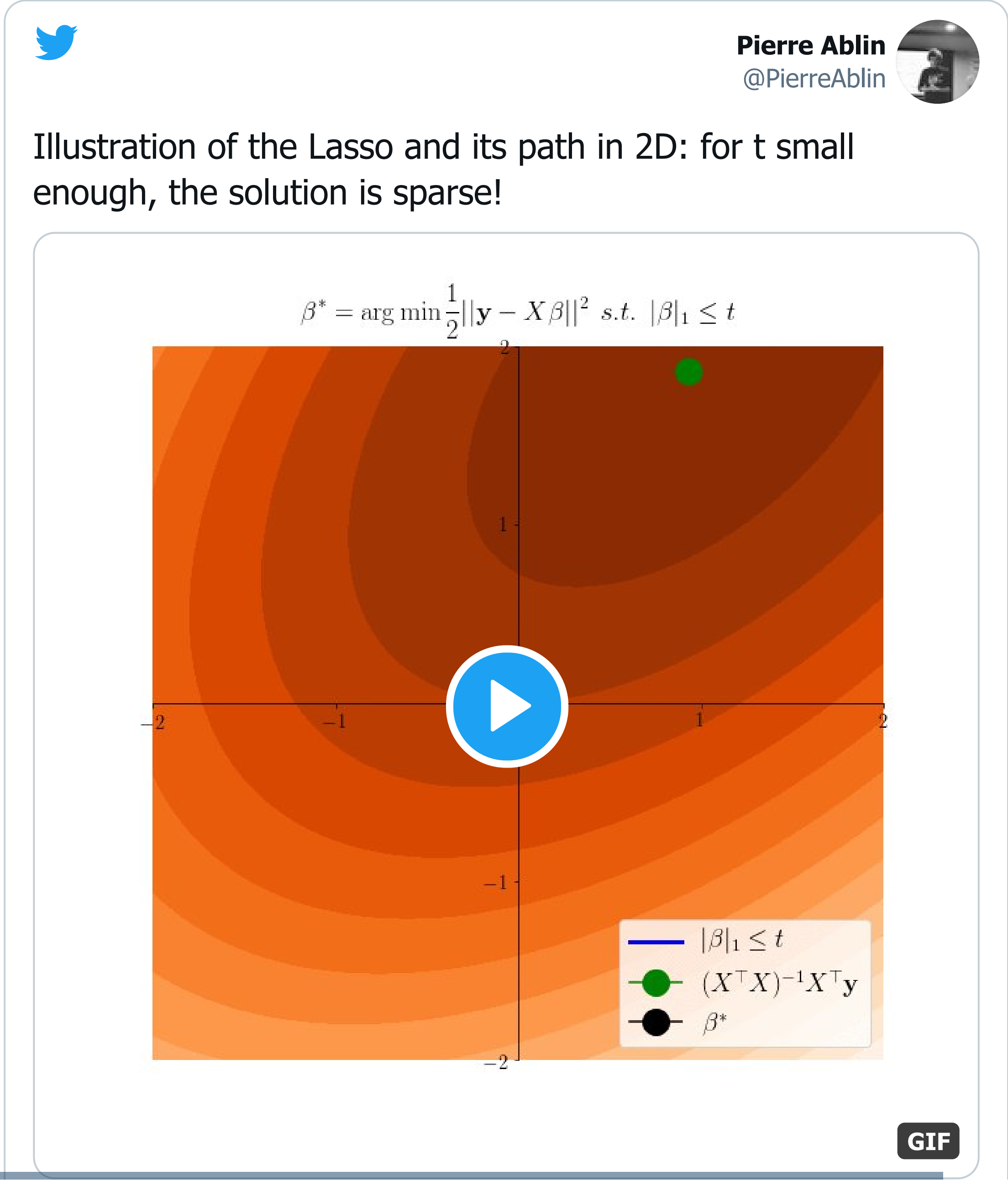
Any penalty that involves the ℓ_1 norm will do this.

Lasso vs. ridge

Contours of the error and constraint functions for lasso (left) and ridge (right). The solid blue areas are the constraint regions, $\beta_1^2 + \beta_2^2 \leq t$, and $|\beta_1| + |\beta_2| \leq t$, while the red ellipses are the contours of the RSS and $\widehat{\beta}$ is the OLS estimator.



Source: James et al. (2017)



Pierre Ablin
@PierreAblin

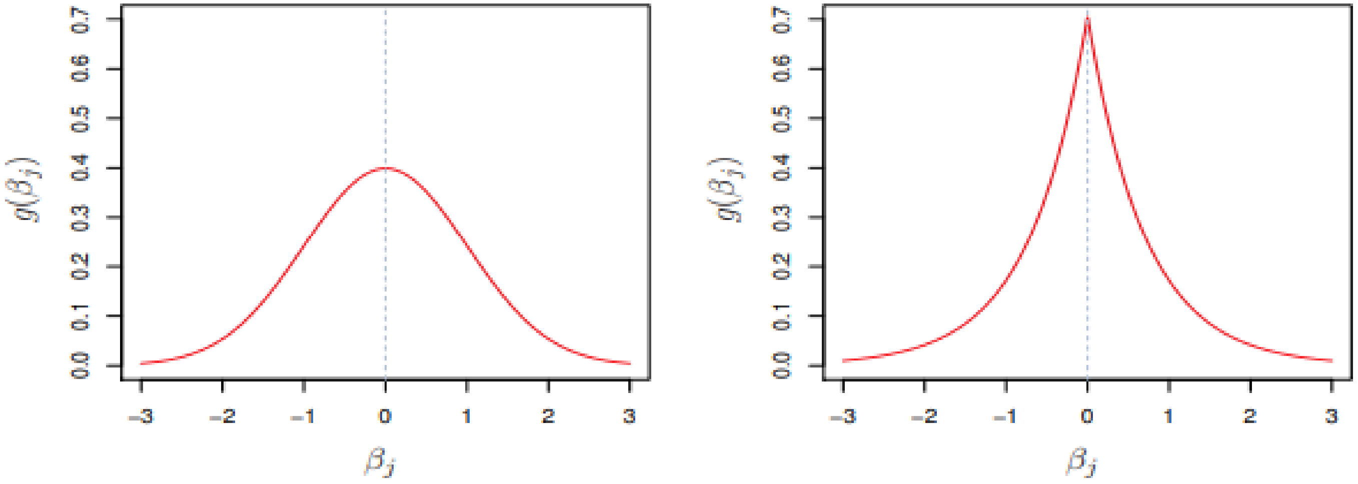
Illustration of the Lasso and its path in 2D: for t small enough, the solution is sparse!

$\beta^* = \arg\min \frac{1}{2}||\mathbf{y} - X\beta||^2 \ s.t. \ |\beta|_1 \leq t$
2
1
−1
−2
−2
−1
1
2
$|\beta|_1 \leq t$
$(X^\top X)^{-1}X^\top \mathbf{y}$
β^*
GIF

Bayesian interpretation of lasso

The prior distribution of $\boldsymbol{\beta}$ under lasso is the double exponential (Laplace) with density $1/2\tau\exp(-|\boldsymbol{\beta}|/\tau)$, where $\tau = 1/\lambda$. The posterior mode is equal to the lasso solution

On the left, normal prior (ridge). On the right, Laplace prior (lasso):

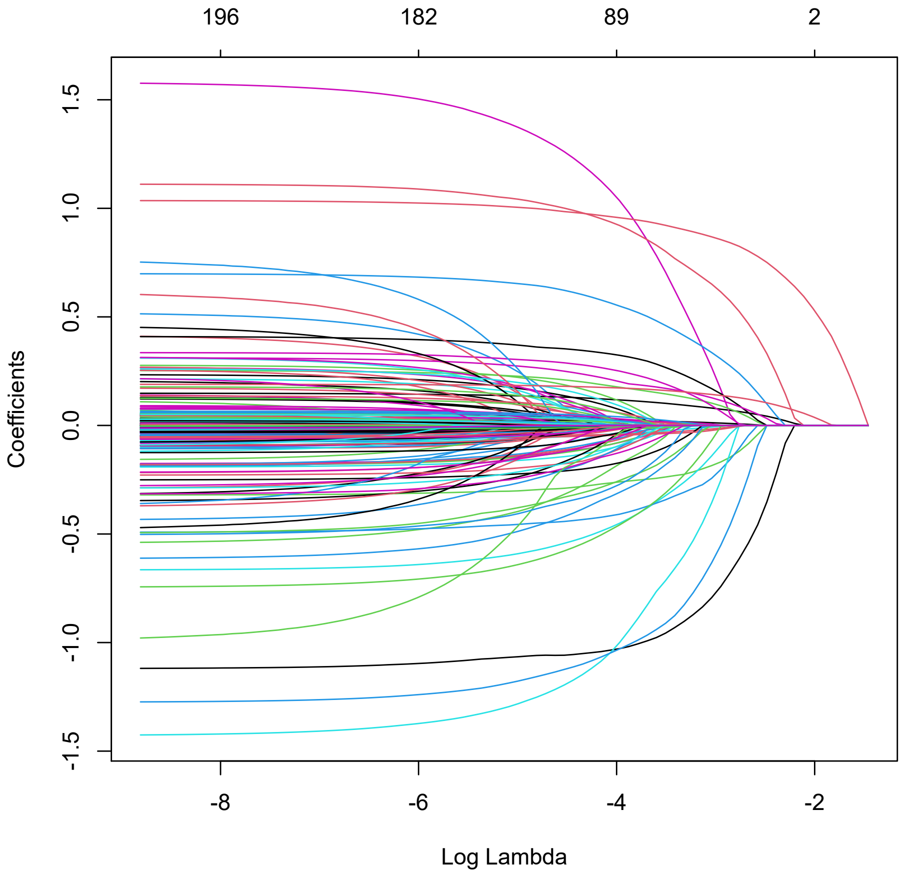


Source: James et al. (2017)

Estimating lasso using glmnet

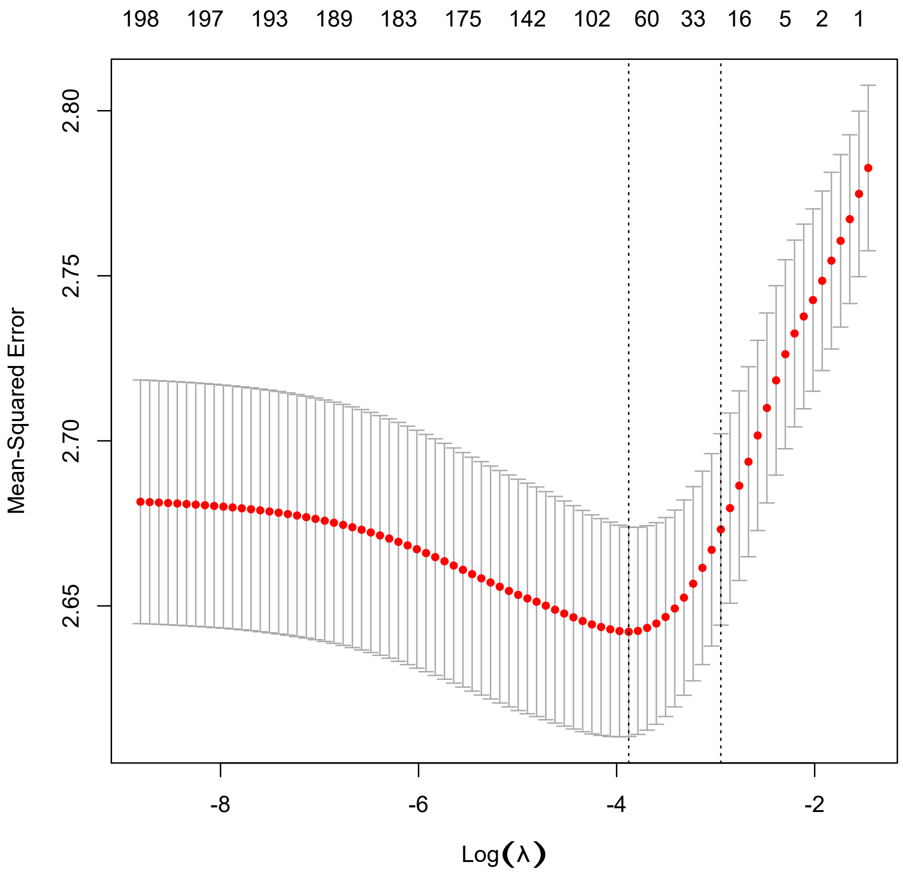
The glmnet() function with alpha = 1 (the default) estimates the entire lasso regularization path.

```
fit_lasso <- glmnet(
  x = X_browser,
  y = Y_browser,
  alpha = 1
)
plot(fit_lasso, xvar = "lambda")
```



Tuning λ

```r
cv_lasso <- cv.glmnet(x = X_browser, y = Y_browser,
plot(cv_lasso, xvar = "lambda")
```



Which features were selected?

Using `s = lambda.min`:

```
coef(cv_lasso, s = "lambda.min") %>%
  tidy() %>%
  as_tibble()
```

```
## # A tibble: 72 x 3
##    row             column    value
##    <chr>           <chr>     <dbl>
##  1 (Intercept)     1       6.02
##  2 65.115.67.11    1       0.00247
##  3 about.com       1      -0.0683
##  4 active.com      1       0.0581
##  5 adoutput.com    1      -0.0305
##  6 adrevolver.com  1      -0.0352
##  7 adserver.com    1       0.0156
##  8 adsonar.com     1       0.0249
##  9 advertising.com 1      -0.0104
## 10 ajc.com         1      -0.0664
## # ... with 62 more rows
```

Using `s = lamda.1se`:

```
coef(cv_lasso, s = "lambda.1se") %>%
  tidy() %>%
  as_tibble()
```

```
## # A tibble: 26 x 3
##    row             column    value
##    <chr>           <chr>     <dbl>
##  1 (Intercept)     1       6.04
##  2 adrevolver.com  1      -0.00384
##  3 amazon.com      1       0.0749
##  4 atomz.com       1       0.165
##  5 azjmp.com       1      -0.168
##  6 bestbuy.com     1       0.0906
##  7 cheaptickets.com 1      0.306
##  8 checkm8.com     1      -0.0912
##  9 circuitcity.com 1       0.146
## 10 citysearch.com  1       0.615
## # ... with 16 more rows
```

A note about shrinkage

Assume that $n = p$ and $\mathbf{X}$ is an $n \times p$ orthonormal matrix with unit vectors and that we are estimating a regression without an intercept.

Under these assumptions, OLS finds $\beta_1, \ldots, \beta_p$ that minimize $\sum_{j=1}^{p} \left(y_j - \beta_j\right)^2$. The solution is $\hat{\beta}_j^{\text{OLS}} = y_j$ (that is, a perfect fit, $R^2 = 1$.)

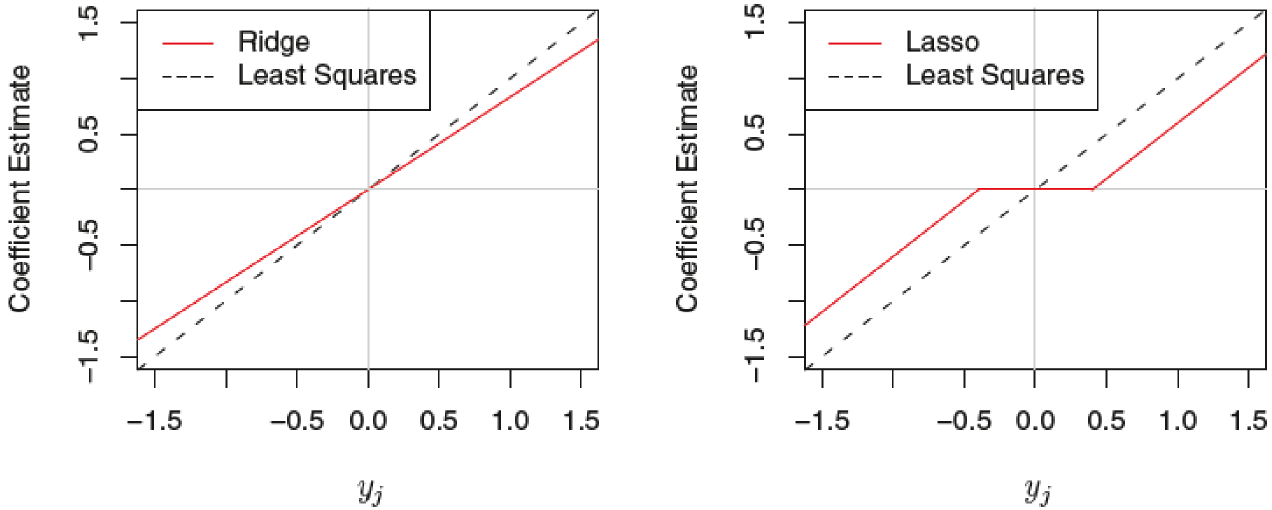
Similarly, ridge and lasso estimates take the form

$$\hat{\beta}_j^{Ridge} = \frac{\hat{\beta}^{\text{OLS}}}{(1+\lambda)}, \qquad \hat{\beta}_j^{Lasso} = \begin{cases} \hat{\beta}^{\text{OLS}} - \lambda/2 & \text{if } \hat{\beta}^{\text{OLS}} > \lambda/2 \\ \hat{\beta}^{\text{OLS}} + \lambda/2 & \text{if } \hat{\beta}^{\text{OLS}} < -\lambda/2 \\ 0 & \text{if } \left|\hat{\beta}^{\text{OLS}}\right| \leq \lambda/2 \end{cases}$$

and best-subset selection drops all variables with coefficients smaller than the t^{th} largest.

Shrinkage and its consequences for econometric analysis

Ridge shrinks proportionally, whereas lasso shrinks by a similar amount, and sufficiently small coefficients are shrunken all the way to zero. The lasso type of shrinkage is called "soft-thresholding."



Source: James et al. (2017)

Main takeaway: We can't just use ridge/lasso coefficients in subsequent econometric analysis (unless you're a Bayesian...)

Dimensionality reduction

A different approach

MAIN IDEA: Instead of selecting and/or shrinking $\boldsymbol{\beta}$, try to find M linear combinations of the x 's, such that $M < p$ and use them to predict y.

Motivating example

Consider the following regression model:

$$y_i = \beta_1 x_{i1} + \beta_2 x_{i2} + \varepsilon_i$$

next, define a new variable, f_i, as a linear combination of x_{i1}, x_{i2}:

$$f_i = \lambda_1 x_{i1} + \lambda_2 x_{i2}$$

for some constants λ_1 and λ_2. This "factor" is a lower dimension representation of the feature space (one variable instead of two.)

According to the dimension reduction approach, it might be the case that

$$\hat{y}_i = \hat{\alpha} f_i$$

(estimated by, say, OLS) would provide better predictions.

How does this relate to penalized regression?

Since

$$\hat{y}_i = \hat{\alpha} f_i = \hat{\alpha}\lambda_1 x_{i1} + \hat{\alpha}\lambda_2 x_{i2},$$

dimension reduction can be viewed as a method that puts a constraint on the estimated β's, i.e.,

$$y_i = \beta_1^* x_{i1} + \beta_2^* x_{i2} + \varepsilon_i$$

where

$$\beta_1^* = \alpha\lambda_1 \qquad \text{and} \qquad \beta_2^* = \alpha\lambda_2$$

The main challenges here:

1. How to choose (estimate) λ_1, λ_2?
2. How many factors to use? (in this case, since $p = 2$, M is at most 2.)

The general case

- Let $f_1, f_2, \ldots, f_p$ represent p linear combinations of the features $x_1, x_2, \ldots, x_p$, such that

$$f_{im} = \lambda_{m1} x_{1i} + \ldots + \lambda_{mp} x_{ip}, \qquad \text{for } m = 1, \ldots, p$$

- The linear model is given by

$$y_i = \alpha_0 + \sum_{m=1}^{M} \alpha_m f_{im} + \varepsilon_i, \qquad \text{for } i = 1, \ldots, N$$

  where $M < p$, and is estimated by OLS.

- The main issue is estimating and selecting the number of factors in a way that improves prediction accuracy (i.e., reduce variance with respect to OLS.)

Principal components regression (PCR)

Let $\mathbf{x} = (x_1, \ldots, x_p)$ and $\boldsymbol{\lambda}_1 = (\lambda_{k1}, \ldots, \lambda_{kp})$ denote a vector of features and scalars, respectively. Principal component analysis (PCA) amounts to:

(1) Find a linear combinations $\boldsymbol{\lambda}_1$ of the elements of $\mathbf{x}$ that maximizes its variance.

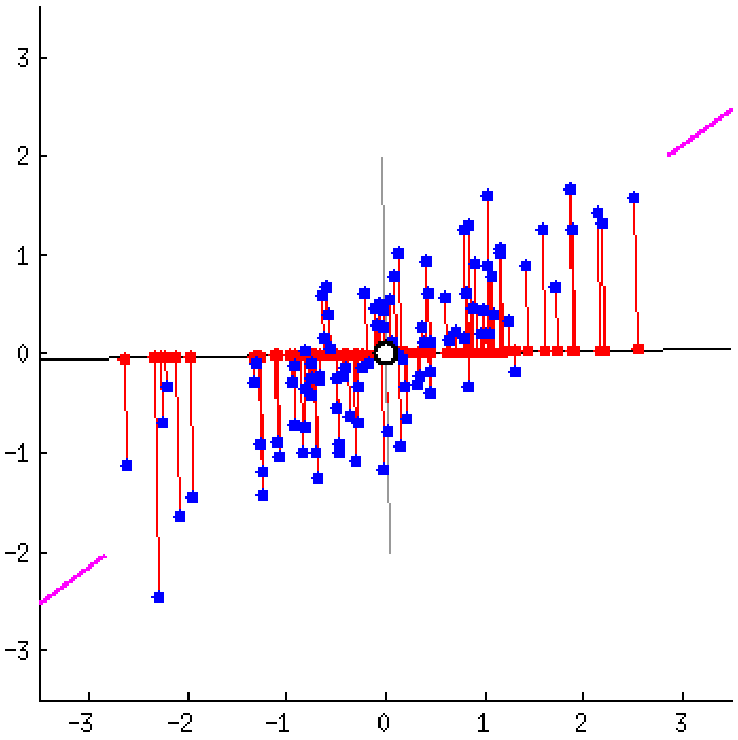
(2) Next, find a linear combination $\boldsymbol{\lambda}_2$ of the elements of $\mathbf{x}$, uncorrelated with $\boldsymbol{\lambda}'_{\mathbf{1}}\mathbf{x}$ having maximum variance.

$$\vdots$$

(p) Finally, find a linear combination $\boldsymbol{\lambda}_p$ of the elements of $\mathbf{x}$, uncorrelated with $\boldsymbol{\lambda}'_{\mathbf{1}}\mathbf{x}, \ldots, \boldsymbol{\lambda}'_{\boldsymbol{p-11}}\mathbf{x}$ having maximum variance.

> $\boldsymbol{\lambda}'_{\boldsymbol{k}}\mathbf{x}$ is the k^{th} **principal component** (PC), and $\boldsymbol{\lambda}_{\boldsymbol{k}}$ is the vector of coefficients or **loadings** for the k^{th} PC.

Illustration in 2-D



Source

Illustration: NBC TV pilots data

- To illustrate, we will use Matt Taddy's NBC pilots dataset that include survey responses for focus groups on TV show pilots as well as the first year of ratings results.

- Our objective: predict viewer interest from pilot surveys, thus helping the studios to make better programming decisions.

- The survey data include 6241 views and 20 questions for 40 shows.

- PE is our outcome of interest. It measures viewers engagement with the show which is reported on a 0 (no attention) to 100 (fully engaged) scale.

Read shows dataset

shows includes data on 40 TV shows.

```
shows <- here("05-regression-regularization/data", "nbc_showdetails.csv") %>%
  read_csv()

head(shows)
```

```
## # A tibble: 6 x 6
##   Show                               Network    PE   GRP Genre           Duration
##   <chr>                              <chr>   <dbl> <dbl> <chr>              <dbl>
## 1 Living with Ed                     HGTV     54   151   Reality               30
## 2 Monarch Cove                       LIFE     64.6 376.  Drama/Adventure       60
## 3 Top Chef                           BRAVO    78.6 808.  Reality               60
## 4 Iron Chef America                  FOOD     62.6  17.3 Reality               30
## 5 Trading Spaces: All Stars          TLC      56    44.1 Reality               60
## 6 Lisa Williams: Life Among the Dead LIFE     56.2 383.  Reality               60
```

Read survey results dataset

The survey dataset includes 6241 views and 20 questions for 40 shows.

```r
survey <- here("05-regression-regularization/data", "nbc_pilotsurvey.csv") %>%
  read_csv()

head(survey)
```

```
## # A tibble: 6 x 22
##   Viewer Show  Q1_Attentive Q1_Excited Q1_Happy Q1_Engaged Q1_Curious Q1_Motivated
##    <dbl> <chr>        <dbl>      <dbl>    <dbl>      <dbl>      <dbl>        <dbl>
## 1     71 Iron~            3          4        4          3          5            4
## 2     71 Trad~            4          4        3          4          5            2
## 3     71 Hous~            4          4        4          5          5            3
## 4     71 What~            4          3        3          3          4            2
## 5     71 Amer~            4          4        3          4          4            4
## 6     73 Next             2          4        2          4          2            3
## # ... with 14 more variables: Q1_Comforted <dbl>, Q1_Annoyed <dbl>,
## #   Q1_Indifferent <dbl>, Q2_Relatable <dbl>, Q2_Funny <dbl>, Q2_Confusing <dbl>,
## #   Q2_Predictable <dbl>, Q2_Entertaining <dbl>, Q2_Fantasy <dbl>,
## #   Q2_Original <dbl>, Q2_Believable <dbl>, Q2_Boring <dbl>, Q2_Dramatic <dbl>,
## #   Q2_Suspenseful <dbl>
```

Aggregate survey data

We now aggregate viewer answers by show (we will use `mean` an the aggregating function)

```
survey_mean <- survey %>%
  select(-Viewer) %>%
  group_by(Show) %>%
  summarise_all(list(mean))

head(survey_mean)
```

```
## # A tibble: 6 x 21
##   Show  Q1_Attentive Q1_Excited Q1_Happy Q1_Engaged Q1_Curious Q1_Motivated
##   <chr>        <dbl>      <dbl>    <dbl>      <dbl>      <dbl>        <dbl>
## 1 30 R~         3.69       3.47     3.65       3.74       3.58         2.96
## 2 Amer~         3.84       3.69     3.58       3.84       3.71         3.32
## 3 Amer~         3.67       3.50     3.46       3.58       3.71         3.01
## 4 Bones         4.12       3.71     3.42       4.06       4.15         3.29
## 5 Clos~         3.80       3.32     3.16       3.83       3.73         3.18
## 6 Cold~         4.05       3.53     3.15       3.91       3.96         3.06
## # ... with 14 more variables: Q1_Comforted <dbl>, Q1_Annoyed <dbl>,
## #   Q1_Indifferent <dbl>, Q2_Relatable <dbl>, Q2_Funny <dbl>, Q2_Confusing <dbl>,
## #   Q2_Predictable <dbl>, Q2_Entertaining <dbl>, Q2_Fantasy <dbl>,
## #   Q2_Original <dbl>, Q2_Believable <dbl>, Q2_Boring <dbl>, Q2_Dramatic <dbl>,
## #   Q2_Suspenseful <dbl>
```

Join `shows` and `survey` to a single tibble

We now generate a tibble that holds `PE` and the mean survey answers, as well as `Genre` and Show.

```
df_join <- shows %>%
  left_join(survey_mean) %>%
  select(PE, starts_with("Q"), Genre, Show)

head(df_join)
```

```
## # A tibble: 6 x 23
##      PE Q1_Attentive Q1_Excited Q1_Happy Q1_Engaged Q1_Curious Q1_Motivated
##   <dbl>        <dbl>      <dbl>    <dbl>      <dbl>      <dbl>        <dbl>
## 1  54           3.89       3.78     3.93       3.87       3.80         3.61
## 2  64.6         4.05       3.86     3.83       3.88       4            3.94
## 3  78.6         3.85       3.60     3.63       3.77       3.86         3.20
## 4  62.6         3.91       3.69     3.61       3.85       3.94         3.33
## 5  56           3.81       3.54     3.51       3.78       3.91         3.29
## 6  56.2         3.72       3.65     3.55       3.66       3.76         3.57
## # ... with 16 more variables: Q1_Comforted <dbl>, Q1_Annoyed <dbl>,
## #   Q1_Indifferent <dbl>, Q2_Relatable <dbl>, Q2_Funny <dbl>, Q2_Confusing <dbl>,
## #   Q2_Predictable <dbl>, Q2_Entertaining <dbl>, Q2_Fantasy <dbl>,
## #   Q2_Original <dbl>, Q2_Believable <dbl>, Q2_Boring <dbl>, Q2_Dramatic <dbl>,
## #   Q2_Suspenseful <dbl>, Genre <chr>, Show <chr>
```

Set up the data for PCR

Features matrix (mean survey answers):

```r
X <- df_join %>%
  select(starts_with("Q")) %>%
  as.matrix()
```

Principal components based on `X` (scaled):

```r
PC <- X %>%
  prcomp(scale. = TRUE) %>%
  predict()
```

where we've used the `prcomp()` and `predict()` function to extract the PCs. Setting `scale.=TRUE` makes sure that `X` is scaled prior to running PCA.

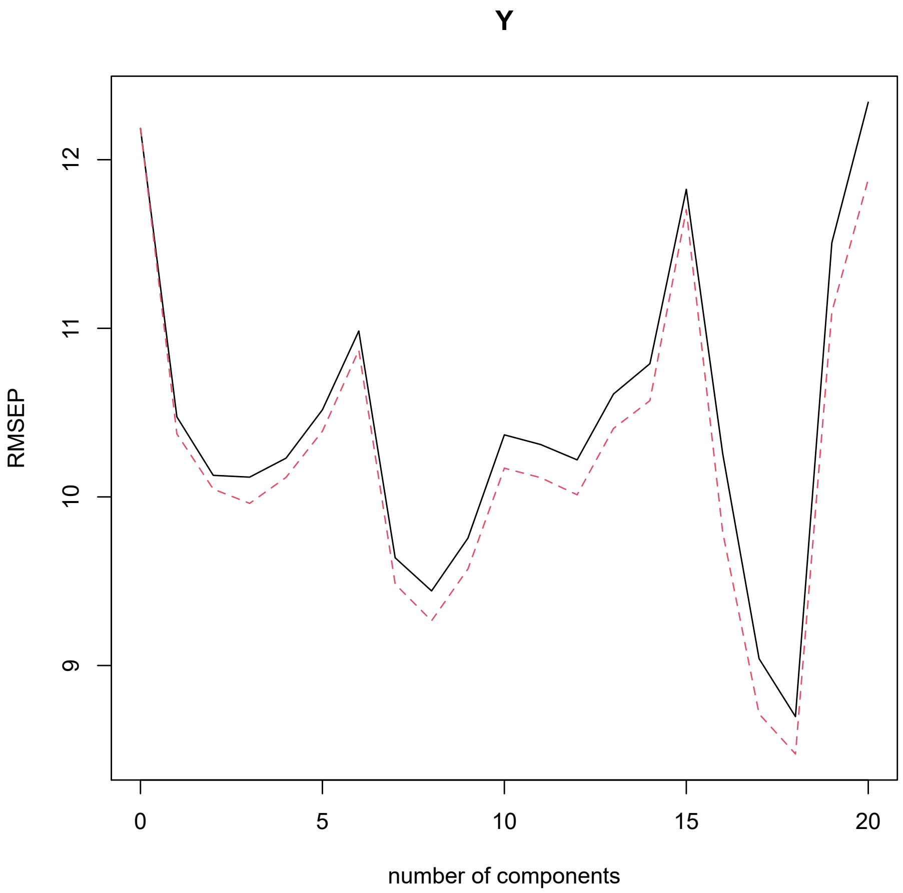
Response variable (`PE`):

```r
Y <- df_join %>%
  select(PE) %>%
  as.matrix()
```

Run PCR using the `pls` package

There are a bunch of packages that perform PCR however, I find that the `{pls}` package is perhaps the most convenient choice (other than using `{tidymodels}`.)

```r
cv_pcr <- pcr(
  Y ~ X,
  scale = TRUE,
  validation = "CV",
  segments = 10
)
validationplot(cv_pcr)
```



Wait, But what is the meaning of these "components"?

Examining the loadings matrix (also known as "rotation matrix") can help us to put label on PCs.

First, we will apply the `prcomp()` function again the the feature matrix and store its output in `pca`.

```
pca <- X %>% prcomp(scale. = TRUE)
```

Then, we can access the rotation matrix using the `$rotation` handle (we'll only look at the first two components.)

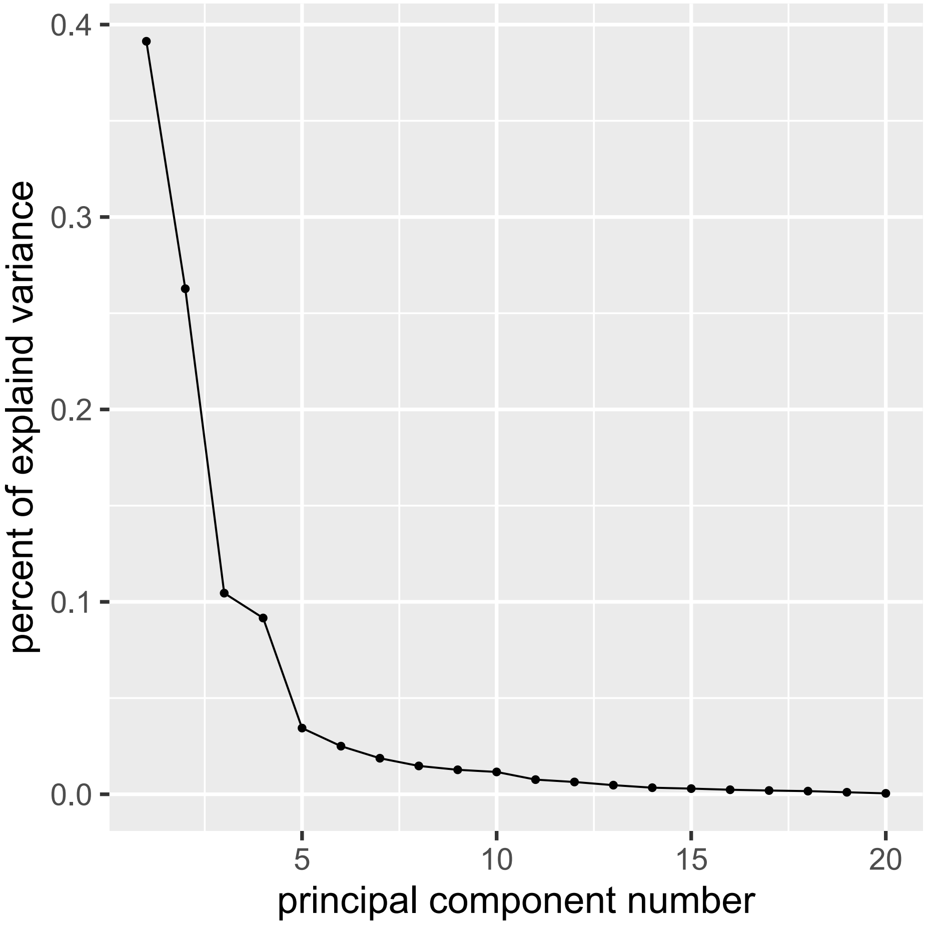
```
pca$rotation[, 1:2] %>% round(1)
```

```
##                  PC1  PC2
## Q1_Attentive    -0.3  0.0
## Q1_Excited      -0.3  0.1
## Q1_Happy        -0.1  0.2
## Q1_Engaged      -0.3  0.0
## Q1_Curious      -0.3  0.0
## Q1_Motivated    -0.2  0.3
## Q1_Comforted    -0.1  0.4
## Q1_Annoyed       0.2  0.3
## Q1_Indifferent   0.2  0.4
## Q2_Relatable    -0.1  0.3
## Q2_Funny         0.1  0.2
## Q2_Confusing    -0.1  0.3
## Q2_Predictable   0.2  0.3
## Q2_Entertaining -0.3 -0.1
## Q2_Fantasy      -0.1  0.2
## Q2_Original     -0.3  0.1
## Q2_Believable   -0.1  0.1
## Q2_Boring        0.2  0.4
## Q2_Dramatic     -0.2  0.0
## Q2_Suspenseful  -0.3  0.0
```

Explained Variance (scree plot)

And here is the cumulative variance explained by each PC ("scree plot")

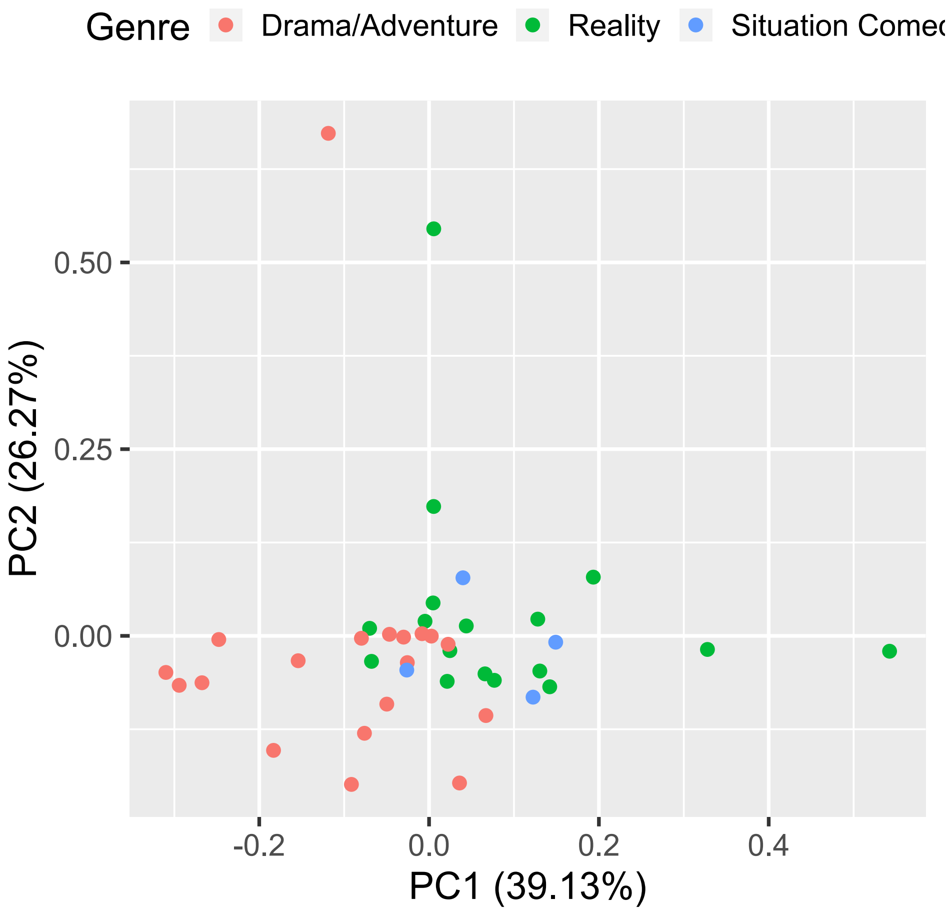
```
pca %>%
  tidy("pcs") %>%
  ggplot(aes(PC, percent)) +
  geom_line() +
  geom_point() +
  labs(
    y = "percent of explaind variance",
    x = "principal component number"
  )
```



Plot data with PCs on axis

The following code plots the survey data based on the first two principal components

```
pca %>%
  autoplot(
    data = df_join,
    colour = "Genre",
    size = 3
  ) +
  theme(legend.position = "top")
```



Notes about PCR, ridge, and lasso

- PCR is an unsupervised method. PCs are extracted without using y. (Can we first extract PCs and *then* proceed to cross validation? Why?)

- PCR implicitly assumes that that data are *dense*, as opposed to lasso where the data is assumed to be *sparse*.

- (Very wonkish:) Ridge regression can also be interpreted as an algorithm that shrinks PCs of x having small variance. By contrast, PCR leaves M PCs untouched and removes the rest.[*]

[*] For further details, see Hastie et al. (2009), Section 3.4.1.

Sparse PCA (SPCA)

Sparse principal component analysis (Zou, Hastie, and, Tibshirani, 2006) employ lasso type penalties to produce components with many loadings zero.

SPCA can be estimated using the `spca()` function from the `{elasticnet}` package

```r
spc <- spca(
  x = X,
  K = 2,
  type = "predictor",
  sparse = "varnum",
  para = c(10, 10)
)
```

where `type="predictor"` means that the input is a feature matrix (and not, say, a covariance matrix), and where the number of PCs `K` is limited to the first two.

Sparsness on the loadings is imposed by `sparse=varnum` which means that we want to limit the number of non-zero loading parameters and `para = c(4,4)` means that we want that each component will have 4 non-zero loadings.

SPCA results

Here are the loadings of the first two sparse PCs based on the `spca()` algorithm:

```
spc$loadings %>% round(1)
```

We can now proceeds to PCR by tuning `K` and or `para` using cross-validation (not in this lecture.)

```
##                  PC1  PC2
## Q1_Attentive    -0.3  0.0
## Q1_Excited      -0.3  0.0
## Q1_Happy         0.0  0.0
## Q1_Engaged      -0.2  0.0
## Q1_Curious       0.0  0.0
## Q1_Motivated    -0.4  0.0
## Q1_Comforted    -0.1  0.3
## Q1_Annoyed       0.0  0.4
## Q1_Indifferent   0.0  0.5
## Q2_Relatable    -0.2  0.2
## Q2_Funny         0.0  0.2
## Q2_Confusing     0.0  0.0
## Q2_Predictable   0.0  0.4
## Q2_Entertaining  0.0  0.0
## Q2_Fantasy      -0.3  0.2
## Q2_Original     -0.2  0.0
## Q2_Believable    0.0  0.0
## Q2_Boring        0.0  0.4
## Q2_Dramatic     -0.2 -0.1
## Q2_Suspenseful  -0.7 -0.1
```

PCR-lasso

Taddy (2019) suggests that instead of tuning the number of principal components, we could try to tune their number using lasso:

$$\min_{\alpha_0,\alpha} \sum_{i=1}^{N} \left(y_i - \alpha_0 - \sum_{j=1}^{p} \hat{f}_{ij} \alpha_j \right)^2 + \lambda \|\boldsymbol{\alpha}\|_1$$

Tune each method using cross-validation

We will now compare 3 methods:

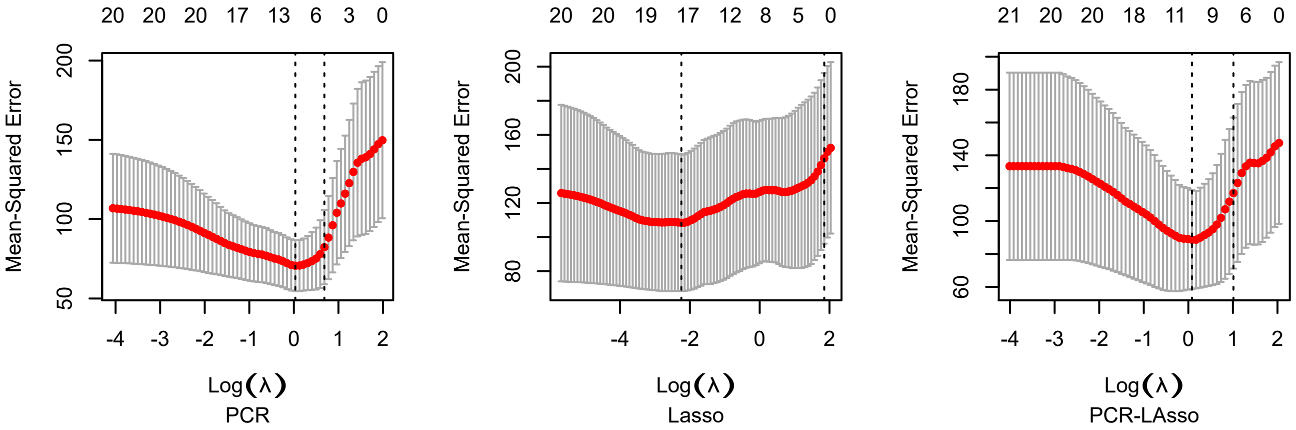
1. PCR-lasso for y on f.
2. Lasso for y on x.
3. PCR-lasso for y onto both x and f.

```r
lasso_pcr  <- cv.glmnet(x = PC, y = Y, nfold = 20)
lasso_x    <- cv.glmnet(x = X , y = Y, nfold = 20)
lasso_pcrx <- cv.glmnet(x = cbind(X, PC) , y = Y, nfold = 20)
```

(Which of the above is preferred depends on the application.)

Plot cross-validation results

```r
par(mfrow=c(1,3))
plot(lasso_pcr, sub = "PCR")
plot(lasso_x, sub = "Lasso")
plot(lasso_pcrx, sub = "PCR-LAsso")
```



Partial least squares (PLS)

- In PCR, PCs are extracted in an unsupervised way - we don't use information on y during the process.

- by contrast, partial least squares (PLS) extracts factors in a supervised way.

- In theory, PLS should have an advantage over PCR, it might pick-up features that are highly correlated with y that PCR would drop.

- In practice, when it comes to prediction, there is little difference in most applications.

The PLS algorithm

For

(1) Regress y onto each x_j independently, i.e.,

$$y_i = \phi_{1j} x_{i,j} + \varepsilon_{ij}, \qquad \text{for } j = 1, \ldots, p$$

and store $\phi_{11}, \ldots, \phi_{1p}$.

(2) Use the estimated vector of coefficients $\hat{\boldsymbol{\phi}}_1 = (\phi_{11}, \ldots, \phi_{1p})$ to construct a new component v_{i1}, which is a linear combination of the xs:

$$v_{i1} = \hat{\boldsymbol{\phi}}_1{}' \mathbf{x}_i$$

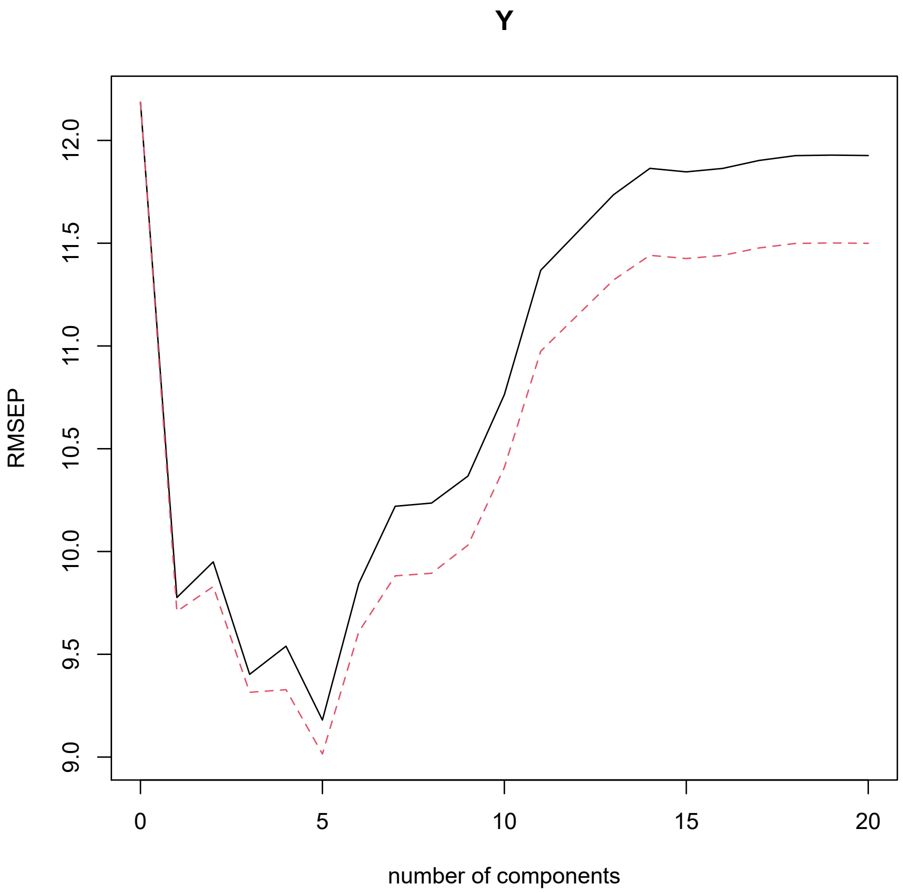
(3) Define the residuals from (1) as a new set of features, and proceed to (2) and estimate the second component, $v_{i2} = \hat{\boldsymbol{\phi}}_2{}' \mathbf{x}_i$.

(4) Continue until you reach the minimum of p or n.

Run PLS using the `pls` package

We're no ready to estimate PLS using, wait for it..., the `{pls}` package!

```
cv_pls <- plsr(
  Y ~ X,
  scale = TRUE,
  validation = "CV",
  segments = 10
)
validationplot(cv_pls)
```



Summary

- *"Despite its simplicity, the linear model has distinct advantages in terms of its interpretability and often shows good predictive performance."* (Hastie and Tibshirani)

- Subset selection is slow, unstable, and infeasible when p is large.

- Ridge provides is fast, works when $p > n$, provides good predictions, but is hard to interpret (keeps all features in.)

- Lasso is fast, works when $p > n$, provides good predictions, and performs variable selection.

- Ridge and lasso shrink the parameters of the model, thus, can't be used out-of-the-box for econometric analysis

- Despite their advantages, linear models need lots of bookkeeping (and memory and computing power) when it comes to explicitly adding non-linearities, interactions, etc.

slides::end()

Source code

References

Hoerl, A. E., and Kennard, R. W. 1970. Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, 12(1), 55-67.

Jolliffe, I. T. (2010). *Principal Component Analysis*. Springer.

Taddy, M. 2019. *Business Data Science: Combining Machine Learning and Economics to Optimize, Automate, and Accelerate Business Decisions* . McGraw-Hill Education. Kindle Edition.

Tibshirani, R. 1996. Regression shrinkage and selection via the lasso. *J . Roy. Statist. Soc. B*, 58(1), 267-288.

Zou, H., Hastie, T., & Tibshirani, R. (2006). Sparse principal component analysis. _Journal of computational and graphical statistic_s, 15(2), 265-286.