Regression and regularization

Lecture 1d

Overview

- Linear regression
- Ridge Regression
- Lasso
- Variable selection

Simple linear regression

Model:

$$y \sim N(w_0 + w_1 x, \sigma^2)$$

or

$$y = w_0 + w_1 x + \epsilon,$$
$$\epsilon \sim N(0, \sigma^2)$$

or

$$p(y|x, w) = N(w_0 + w_1 x, \sigma^2)$$

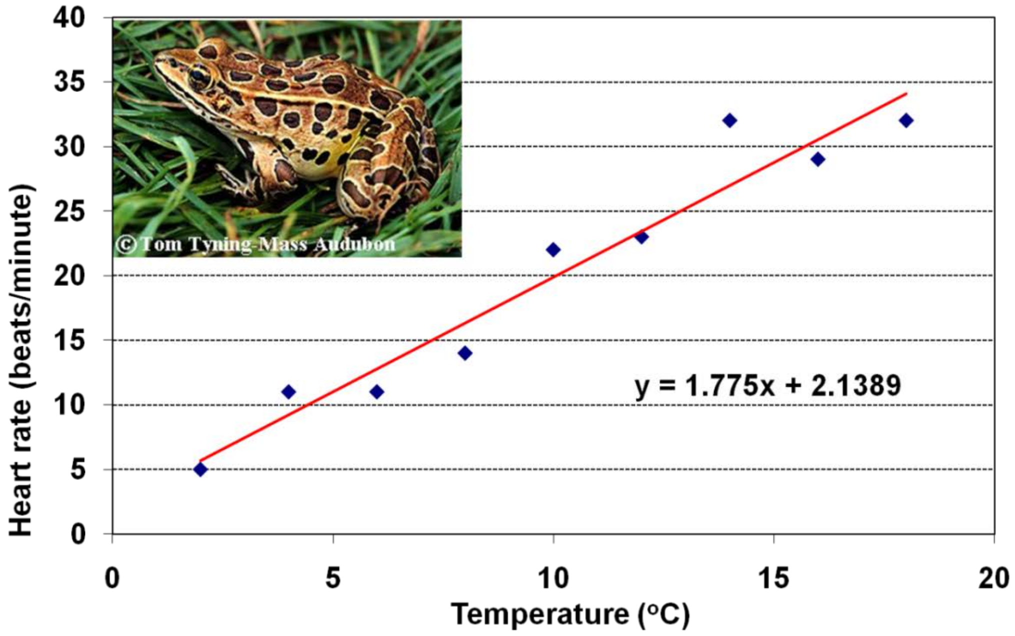
Terminology:

w_0: intercept (or bias)

w_1: regression coefficient

Response

The target responds directly and linearly to changes in the feature



Ordinary least squares regression (OLS)

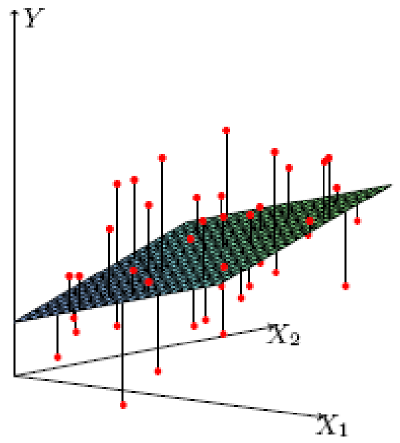
Model:

$$y \sim N(\boldsymbol{w}^T \boldsymbol{x}, \sigma^2)$$

where

$$\boldsymbol{w} = \{w_0, \dots w_d\}$$
$$\boldsymbol{x} = \{1, x_1, \dots x_d\}$$

Why is ”1” here?



The response variable responds directly and linearly to changes in each of the inputs

Ordinary least squares regression

Given data set D

Case	X_1	X_2			X_p	Y
1	x_{11}	x_{21}			x_{p1}	y_1
2	x_{12}	x_{22}			x_{p2}	y_2
3	x_{13}	x_{23}			x_{p3}	y_3
N	x_{1N}	x_{2N}			x_{pN}	y_N

Estimation: maximizing the likelihood

$$\widehat{w} = \max_{\mathrm{w}} p(D|w)$$

Is equivalent to minimizing

$$RSS(w) = \sum_{i=1}^{n} (Y_i - \boldsymbol{w}^T \boldsymbol{X_i})^2$$

Matrix formulation of OLS regression

Optimality condition:

$$\boldsymbol{X}^T(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w}) = 0$$

where

$$\boldsymbol{X} = \begin{pmatrix} 1 & x_{11} & x_{21} & & x_{p1} \\ 1 & x_{12} & x_{22} & & x_{p2} \\ & & & & \\ & & & & \\ 1 & x_{1N} & x_{2N} & & x_{pN} \end{pmatrix}$$

and

$$\boldsymbol{y} = \begin{pmatrix} y_1 \\ y_2 \\ \\ \\ y_N \end{pmatrix}$$

Parameter estimates and predictions

- Least squares estimates of the parameters

$$\hat{w} = (\boldsymbol{X}^T \boldsymbol{X})^{-1} \boldsymbol{X}^T \boldsymbol{y}$$

- Predicted values $\hat{\boldsymbol{y}} = \boldsymbol{X}\hat{w} = \boldsymbol{X}(\boldsymbol{X}^T \boldsymbol{X})^{-1} \boldsymbol{X}^T \boldsymbol{y} = \boldsymbol{P}\boldsymbol{y}$

Hat matrix

Why is it called so?

- Linear regression belongs to the class of **linear smoothers**

Degrees of freedom

Definition:

$$df(\hat{y}) = \frac{1}{\sigma^2}\sum_{i=1}^{N} Cov(\hat{y}_i, y_i)$$

- Larger covariance → stronger connection → model can approximate data better→ model more flexible (complex)
- For linear smoothers $\hat{Y} = S(X)Y$

$$df = trace(S)$$

- For linear regression, degrees of freedom is

$$df = trace(P) = p$$

Different types of features

- Interval variables
- Numerically coded ordinal variables
    - (small=1, medium=2, large=3)
- Dummy coded qualitative variables

Basis function expansion:
If $y = w_0 + w_1 x_1 + w_2 x_1^2 + w_3 e^{-x_2} + \epsilon$,

Model becomes linear if to recompute:

$$\phi_1(x_1) = x_1$$
$$\phi_2(x_1) = x_1^2$$
$$\phi_3(x_1) = e^{-x_2}$$

Example of dummy coding:

$$x_1 = \begin{cases} 1, \text{if Jan} \\ 0, \text{otherwise} \end{cases}$$

$$x_2 = \begin{cases} 1, \text{if Feb} \\ 0, \text{otherwise} \end{cases}$$

.

.

.

$$x_{11} = \begin{cases} 1, \text{if Nov} \\ 0, \text{otherwise} \end{cases}$$

Basis function expansion

- In general $\phi_1(\dots)$ may be a function of several x components
- Having data given by **X,** compute new data
- $\Phi = \begin{pmatrix} 1 & \phi_1(x_{11,} \dots, x_{1p}) & .. & \phi_p(x_{11,} \dots, x_{1p}) \\ & \dots & \dots & \dots \\ 1 & \phi_1(x_{n1,} \dots, x_{np}) & \dots & \phi_p(x_{n1,} \dots, x_{np}) \end{pmatrix}$
- If doing a basis function in a model, replace **X** by Φ everywhere where **X** is used:

$$\hat{\boldsymbol{y}} = \Phi(\Phi^T\Phi)^{-1}\Phi^T\boldsymbol{y}$$

Linear regression in R

- fit=lm(formula, data, subset, weights,…)
  - **data** is the data frame containing the predictors and response values
  - **formula** is expression for the model
  - **subset** which observations to use (training data)?
  - **weights** should weights be used?

fit is object of class **lm** containing various regression results.

- Useful functions (many are generic, used in many other models)
  - Get details about the particular function by ”.”, for ex. predict.lm

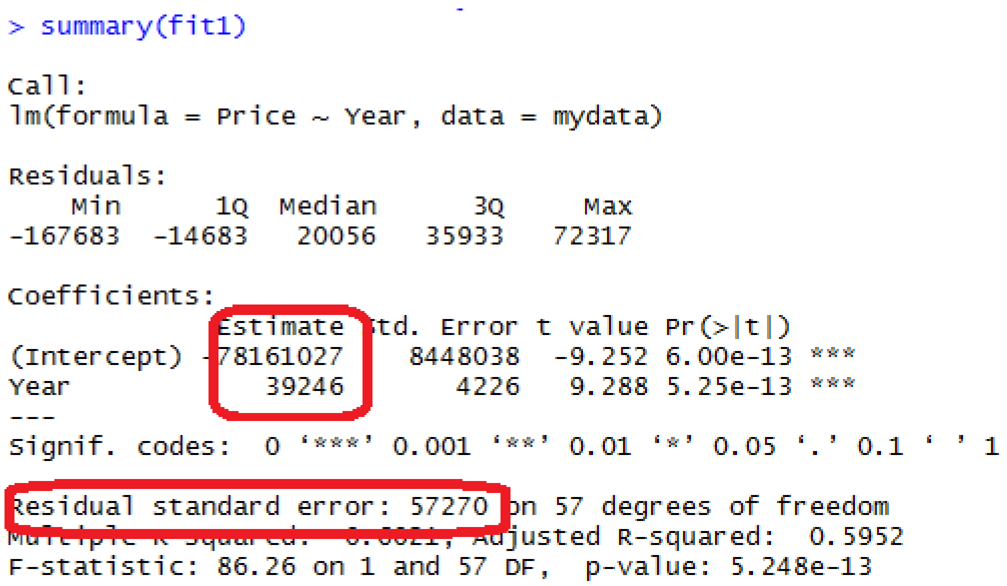
```
summary(fit)
predict(fit, newdata, se.fit, interval)
coefficients(fit) # model coefficients
confint(fit, level=0.95) # CIs for model parameters
fitted(fit) # predicted values
residuals(fit) # residuals
```

An example of ordinary least squares regression

```
mydata=read.csv2("Bilexempel.csv")
fit1=lm(Price~Year, data=mydata)
summary(fit1)
fit2=lm(Price~Year+Mileage+Equipment,
data=mydata)
summary(fit2)
```

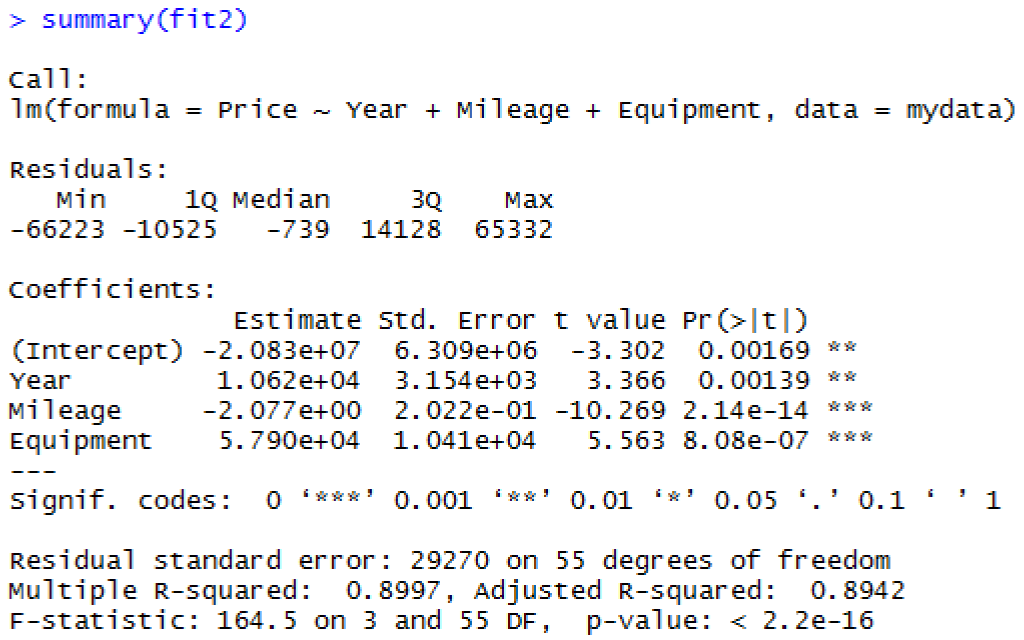
Response variable:
Requested price of used Porsche cars (1000 SEK)

Inputs:
X_1 = Manufacturing year
X_2 = Milage (km)
X_4 = Equipment (0 or 1)



```
> summary(fit1)

Call:
lm(formula = Price ~ Year, data = mydata)

Residuals:
    Min      1Q  Median      3Q     Max 
-167683  -14683   20056   35933   72317 

Coefficients:
             Estimate Std. Error t value Pr(>|t|)    
(Intercept) -78161027    8448038  -9.252 6.00e-13 ***
Year            39246       4226   9.288 5.25e-13 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 57270 on 57 degrees of freedom
Multiple R-squared: 0.6021, Adjusted R-squared:  0.5952 
F-statistic: 86.26 on 1 and 57 DF,  p-value: 5.248e-13
```

An example of ordinary least squares regression

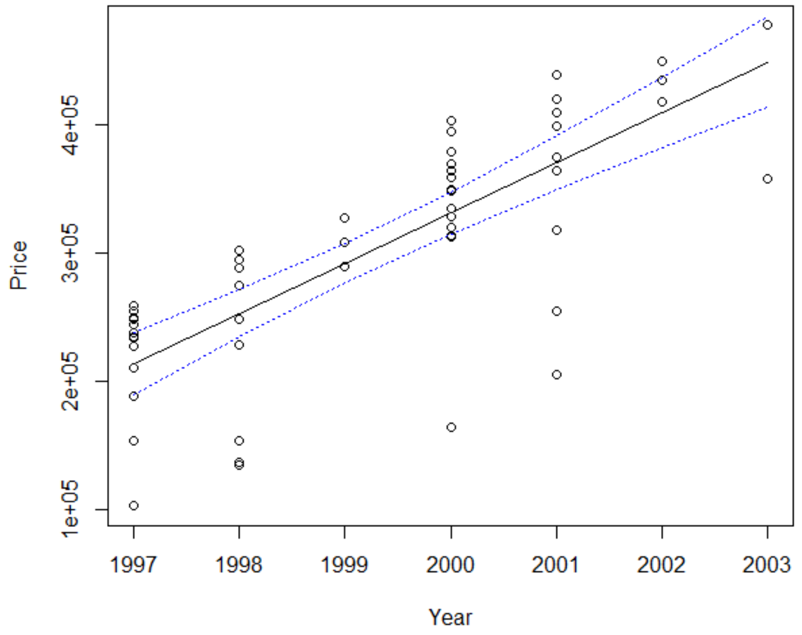


```
> summary(fit2)

Call:
lm(formula = Price ~ Year + Mileage + Equipment, data = mydata)

Residuals:
   Min     1Q Median     3Q    Max 
-66223 -10525   -739  14128  65332 

Coefficients:
              Estimate Std. Error t value Pr(>|t|)    
(Intercept) -2.083e+07  6.309e+06  -3.302  0.00169 ** 
Year         1.062e+04  3.154e+03   3.366  0.00139 ** 
Mileage     -2.077e+00  2.022e-01 -10.269 2.14e-14 ***
Equipment    5.790e+04  1.041e+04   5.563 8.08e-07 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 29270 on 55 degrees of freedom
Multiple R-squared:  0.8997, Adjusted R-squared:  0.8942 
F-statistic: 164.5 on 3 and 55 DF,  p-value: < 2.2e-16
```

An example of ordinary least squares regression

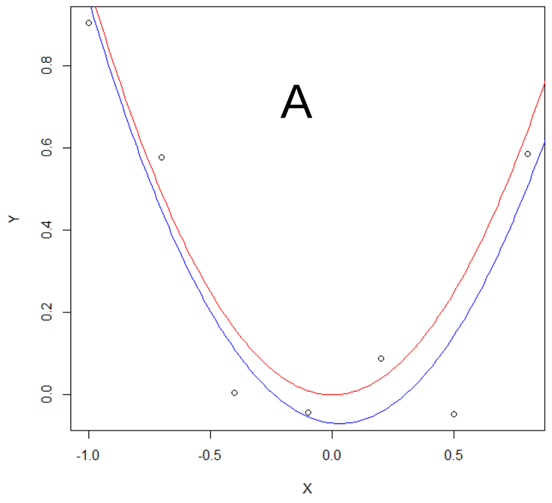
- Prediction

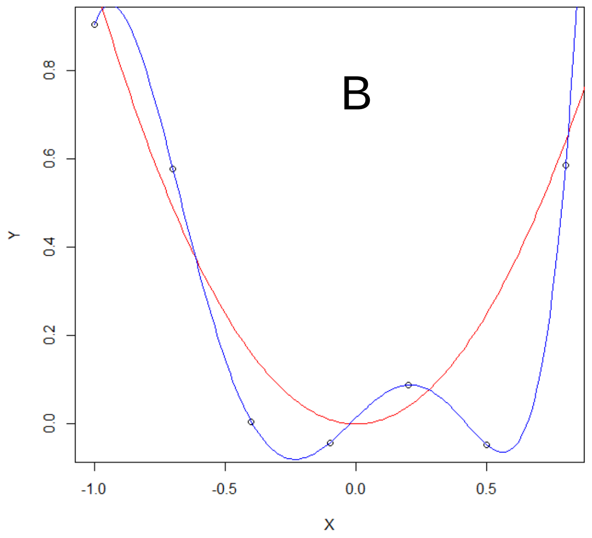
```
fitted <- predict(fit1, interval =
"confidence")

# plot the data and the fitted line
attach(mydata)
plot(Year, Price)
lines(Year, fitted[, "fit"])

# plot the confidence bands
lines(Year, fitted[, "lwr"], lty = "dotted",
col="blue")
lines(Year, fitted[, "upr"], lty = "dotted",
col="blue")
detach(mydata)
```



Ridge regression

- Problem: linear regression can overfit:
  - Take $Y := Y, X_1 = X, X_2 = X^2, \dots, X_p = X^p \rightarrow$ polinomial model, fit by linear regression
  - High degree of polynomial leads to overfitting.





Ridge regression

- **Idea**: Keep all predictors but shrink coefficients to make model less complex

minimize $-loglikelihood + \lambda_0 ||w||_2^2$

→ **l_2 regularization**

  – Given that model is Gaussian, we get **Ridge regression:**

$$\hat{w}^{ridge} = \text{argmin}\left\{ \sum_{i=1}^{N} (y_i - w_0 - w_1 x_{1j} - \ldots - w_p x_{pj})^2 + \lambda \sum_{j=1}^{p} w_j^2 \right\}$$

- $\lambda > 0$ is **penalty factor**

Ridge regression

Equivalent form

$$\hat{w}^{ridge} = \operatorname{argmin} \sum_{i=1}^{N} (y_i - w_0 - w_1 x_{1j} - \ldots - w_p x_{pj})^2$$

subject to $\sum_{j=1}^{p} w_j^2 \le s$

Solution

$$\boxed{\hat{\boldsymbol{w}}^{ridge} = \left(\boldsymbol{X}^T\boldsymbol{X} + \lambda\boldsymbol{I}\right)^{-1}\boldsymbol{X}^T\boldsymbol{y}}$$

$$\hat{\boldsymbol{y}} = \boldsymbol{X}\hat{\boldsymbol{w}} = \boldsymbol{X}(\boldsymbol{X}^T\boldsymbol{X} + \lambda\boldsymbol{I})^{-1}\boldsymbol{X}^T\boldsymbol{y} = \underbrace{\boldsymbol{P}}_{\text{Hat matrix}}\boldsymbol{y}$$

How do we compute degrees of freedom here?

Ridge regression

Properties

- Extreme cases:
  - $\lambda = 0$ usual linear regression (no shrinkage)
  - $\lambda = +\infty$ fitting a constant ($w = 0$ except of w_0)
- When input variables are ortogonal (not realistic), $X^T X = I \rightarrow$

  $\hat{\boldsymbol{w}}^{\text{ridge}} = \frac{1}{1+\lambda} \mathbf{w}^{\mathbf{linreg}}$ → coefficients are equally shrunk
- **Ridge regression is particularly useful if the explanatory variables are strongly correlated to each other.**
  - Correlated variables often correspond large w → shrunk
- Degrees of freedom decrease when λ increases
  - $\lambda = 0 \rightarrow d.f. = p$

Ridge regression

Properties

- Shrinking enables estimation of regression coefficients even if the number of parameters exceeds the number of cases! ($X^T X + \lambda I$ is always nonsingular)
  - Compare with linear regression

- How to estimate λ?
  - cross-validation

Ridge regression

- Bayesian view
  - Ridge regression is just a special form of Bayesian Linear Regression with constant σ^2:

$$\boldsymbol{y} \sim N(\boldsymbol{y} | \boldsymbol{w_o} + \boldsymbol{X}\boldsymbol{w}, \boldsymbol{\sigma^2 I})$$

$$\boldsymbol{w} \sim \boldsymbol{N}\left(\boldsymbol{0}, \frac{\sigma^2}{\lambda}\boldsymbol{I}\right)$$

Theorem MAP estimate to the Bayesian Ridge is equal to solution in frequenist Ridge

$$\widehat{\boldsymbol{w}}^{ridge} = (\boldsymbol{X}^T\boldsymbol{X} + \lambda\boldsymbol{I})^{-1}\boldsymbol{X}^T\boldsymbol{y}$$

- In Bayesian version, we can also make inference about λ

Ridge regression

Example **Computer Hardware Data Set** : performance measured for various processors and also

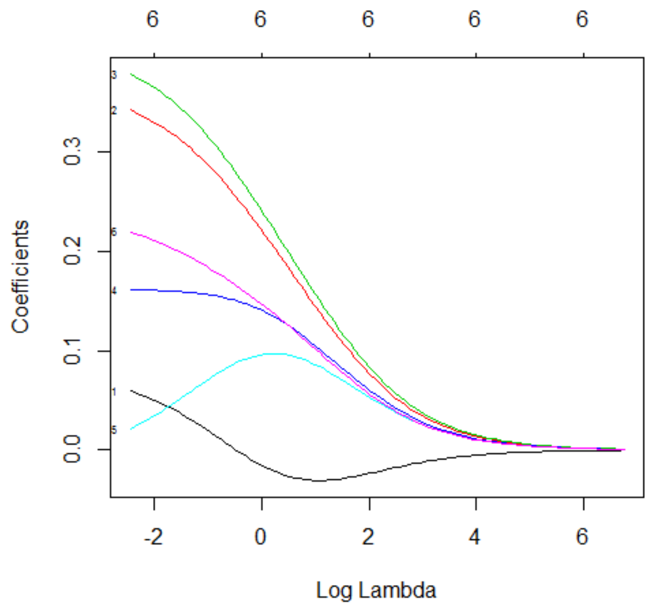
- Cycle time
- Memory
- Channels
- …

Build model predicting performance

Ridge regression

- R code: use package **glmnet** with alpha=0 (Ridge regression)
- Seeing how Ridge converges

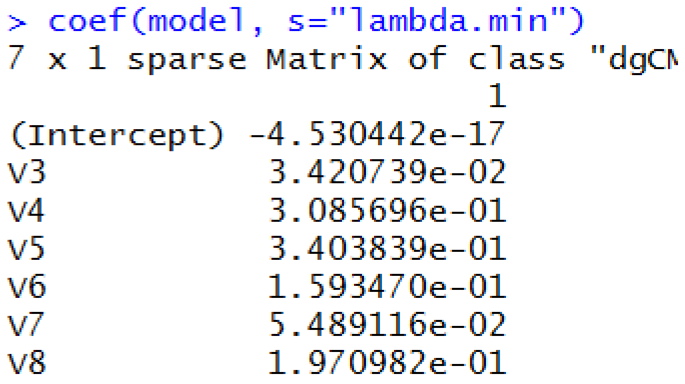
```
data=read.csv("machine.csv", header=F)
covariates=scale(data[,3:8])
response=scale(data[, 9])

model0=glmnet(as.matrix(covariates),
response, alpha=0,family="gaussian")
plot(model0, xvar="lambda", label=TRUE)
```



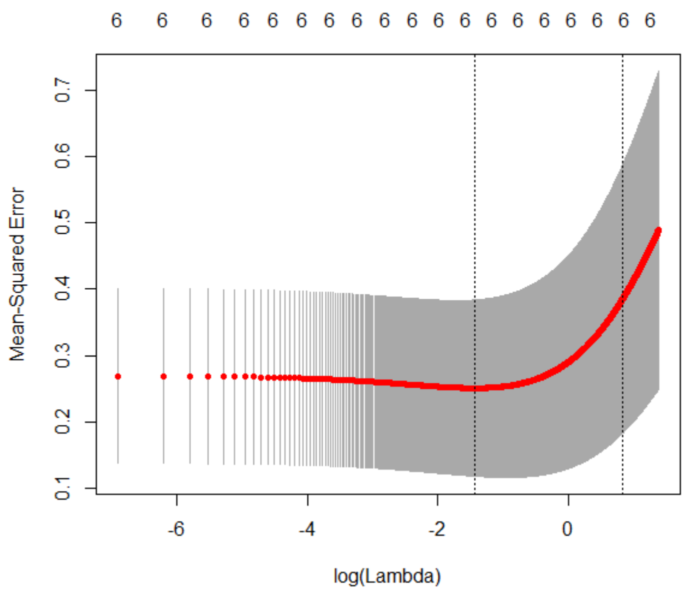
Ridge regression

- Choosing the best model by cross-validation:

```
model=cv.glmnet(as.matrix(covariates),
response, alpha=0,family="gaussian")
model$lambda.min
plot(model)
coef(model, s="lambda.min")
```



```
> coef(model, s="lambda.min")
7 x 1 sparse Matrix of class "dgCM
                        1
(Intercept) -4.530442e-17
V3           3.420739e-02
V4           3.085696e-01
V5           3.403839e-01
V6           1.593470e-01
V7           5.489116e-02
V8           1.970982e-01
```



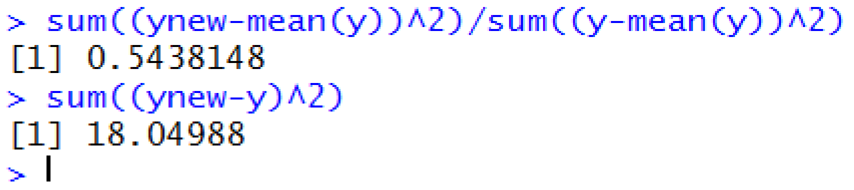
```
> model$lambda.min
 [1] 0.046
```

Ridge regression

- How good is this model in prediction?

```
ind=sample(209, floor(209*0.5))
data1=scale(data[,3:9])
train=data1[ind,]
test=data1[-ind,]

covariates=train[,1:6]
response=train[, 7]
model=cv.glmnet(as.matrix(covariates), response, alpha=1,family="gaussian",
lambda=seq(0,1,0.001))
y=test[,7]
ynew=predict(model, newx=as.matrix(test[, 1:6]), type="response")

#Coefficient of determination
sum((ynew-mean(y))^2)/sum((y-mean(y))^2)

sum((ynew-y)^2)
```

Note that data are so small so numbers change much for other train/test



```
> sum((ynew-mean(y))^2)/sum((y-mean(y))^2)
[1] 0.5438148
> sum((ynew-y)^2)
[1] 18.04988
> |
```

LASSO

- **Idea**: Similar idea to Ridge
- Minimize minus loglikelihood plus **linear** penalty factor→ **l_1 regularization**
  - Given that model is Gaussian, we get **LASSO** (least absolute shrinkage and selection operator):

$$\hat{w}^{lasso} = \operatorname{argmin}\left\{ \sum_{i=1}^{N} (y_i - w_0 - w_1 x_{1j} - \ldots - w_p x_{pj})^2 + \lambda \sum_{j=1}^{p} |w_i| \right\}$$

- $\lambda > 0$ is **penalty factor**

LASSO

- Equivalently

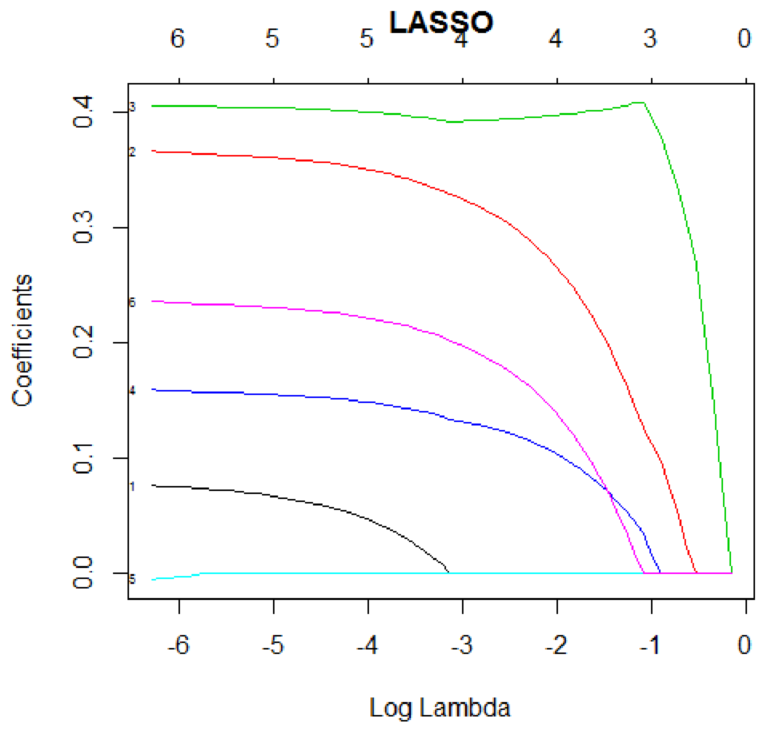
$$\hat{w}^{lasso} = \operatorname{argmin} \sum_{i=1}^{N} (y_i - w_0 - w_1 x_{1j} - \ldots - w_p x_{pj})^2$$

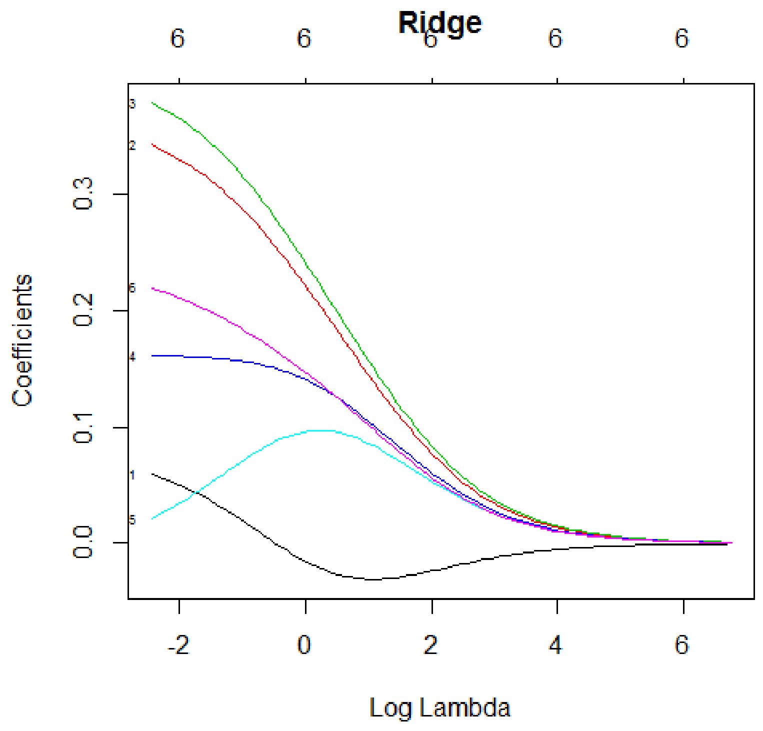
subject to $\sum_{j=1}^{p} |w_i| \leq s$

LASSO vs Ridge

- **LASSO yields sparse solutions!**

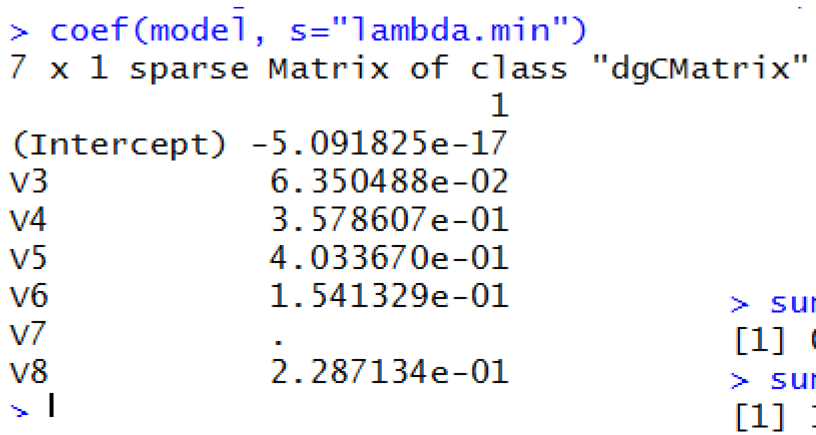
Example Computer hardware data



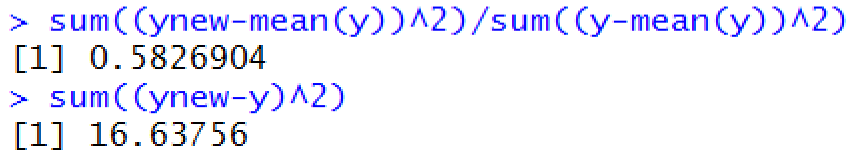


LASSO vs Ridge

- Only 5 variables selected by LASSO



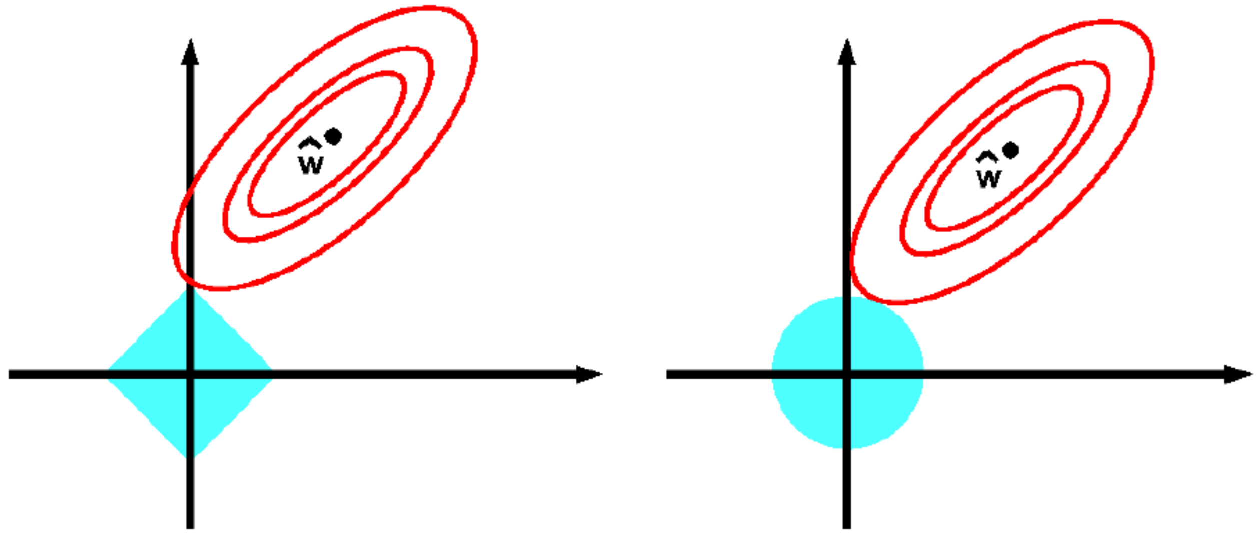
```
> coef(model, s="lambda.min")
7 x 1 sparse Matrix of class "dgCMatrix"
                         1
(Intercept) -5.091825e-17
V3           6.350488e-02
V4           3.578607e-01
V5           4.033670e-01
V6           1.541329e-01
V7           .
V8           2.287134e-01
> 
```



```
> sum((ynew-mean(y))^2)/sum((y-mean(y))^2)
[1] 0.5826904
> sum((ynew-y)^2)
[1] 16.63756
```

LASSO vs Ridge

- Why Lasso leads to sparse solutions?
  - Feasible area for Ridge is a circle (2D)
  - Feasible area for LASSO is a polygon (2D)



LASSO properies

- **Lasso is widely used when $\boldsymbol{p \gg n}$**
  - Linear regression breaks down when $p > n$
  - Application: DNA sequence analysis, Text Prediction

- When inputs are orthonormal,

$$\widehat{w}_i^{\text{lasso}} = sign\left(w_i^{linreg}\right)\left(\left|w_i^{linreg}\right| - \frac{\lambda}{2}\right)_+$$

- No explicit formula for $\widehat{\boldsymbol{w}}^{\boldsymbol{lasso}}$
  - Optimization algorithms used

Coding in R: use glmnet() with alpha=1

Variable selection

- .. Or “Feature selection”

Often, we do not need all features available in the data to be in the model

Reasons:

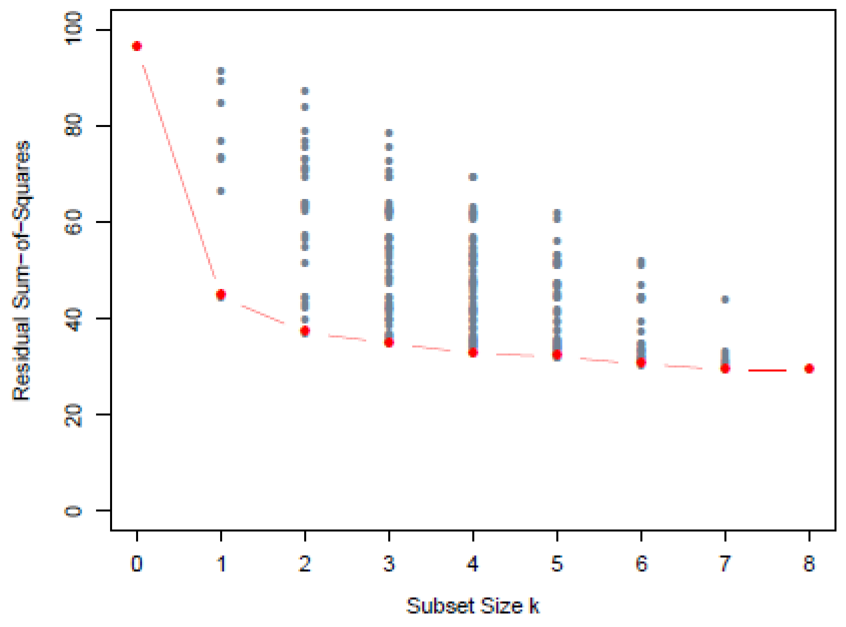
- Model can become overfitted (recall polynomial regression)
- Large number of predictors → model is difficult to use and interpret

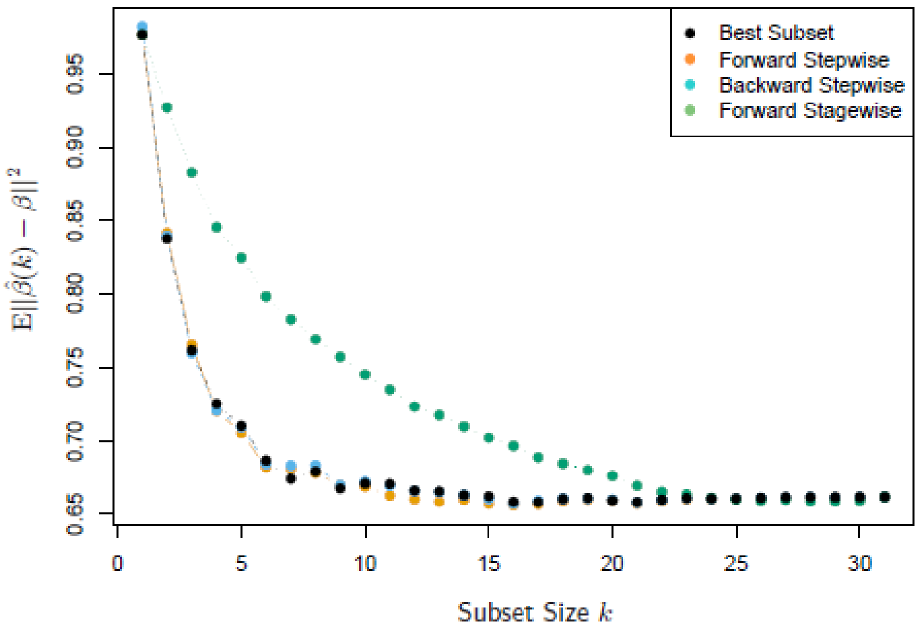
Variable selection

Alternative 1: Variable subset selection

- Best subset selection:
  - Consider different subsets of the full set of features, fit models and evaluate their quality
    - Problem: computationally difficult for p around 30 or more
    - How to choose the best model size? Some measure of predictive performance normally used (ex. AIC).
- Forward and Backward stepwise selection
  - Starts with 0 features (or full set) and then adds a feature (removes feature) that most improves the measure selected.
    - Can handle large *p* quickly
    - Does not examine all possible subsets (not the “best”)

RSS and MSE depend on k

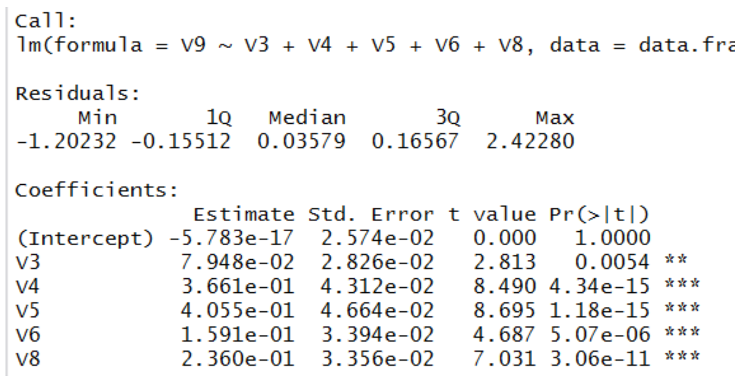




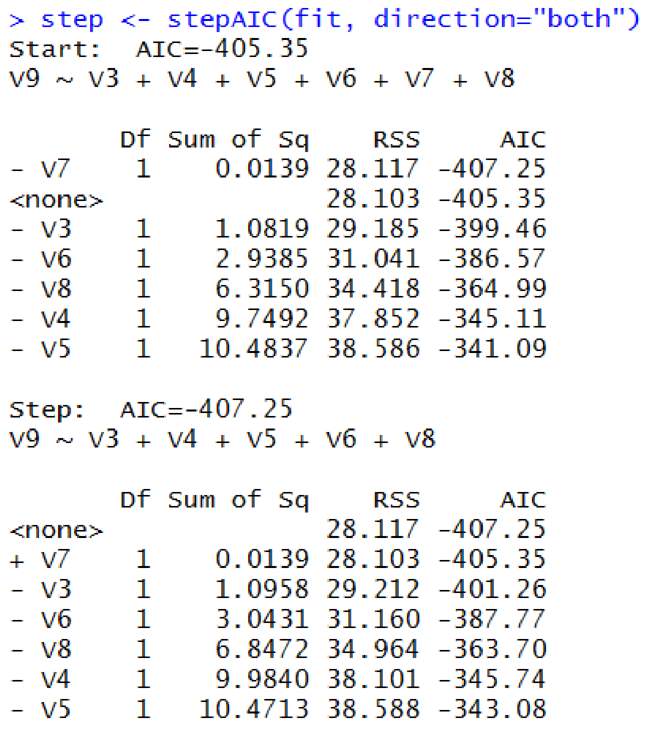
Variable selection in R

- Use stepAIC() in MASS

```
library(MASS)
fit <- lm(V9~.,data=data.frame(data1))
step <- stepAIC(fit, direction="both")
step$anova
summary(step)
```



```
Call:
lm(formula = V9 ~ V3 + V4 + V5 + V6 + V8, data = data.fra

Residuals:
     Min       1Q   Median       3Q      Max
-1.20232 -0.15512  0.03579  0.16567  2.42280

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept) -5.783e-17  2.574e-02   0.000   1.0000
V3           7.948e-02  2.826e-02   2.813   0.0054 **
V4           3.661e-01  4.312e-02   8.490 4.34e-15 ***
V5           4.055e-01  4.664e-02   8.695 1.18e-15 ***
V6           1.591e-01  3.394e-02   4.687 5.07e-06 ***
V8           2.360e-01  3.356e-02   7.031 3.06e-11 ***
```



```
> step <- stepAIC(fit, direction="both")
Start:  AIC=-405.35
V9 ~ V3 + V4 + V5 + V6 + V7 + V8

       Df Sum of Sq    RSS     AIC
- V7    1    0.0139 28.117 -407.25
<none>              28.103 -405.35
- V3    1    1.0819 29.185 -399.46
- V6    1    2.9385 31.041 -386.57
- V8    1    6.3150 34.418 -364.99
- V4    1    9.7492 37.852 -345.11
- V5    1   10.4837 38.586 -341.09

Step:  AIC=-407.25
V9 ~ V3 + V4 + V5 + V6 + V8

       Df Sum of Sq    RSS     AIC
<none>              28.117 -407.25
+ V7    1    0.0139 28.103 -405.35
- V3    1    1.0958 29.212 -401.26
- V6    1    3.0431 31.160 -387.77
- V8    1    6.8472 34.964 -363.70
- V4    1    9.9840 38.101 -345.74
- V5    1   10.4713 38.588 -343.08
```