# Model Selection and Shrinkage

Vahid Partovi Nia

Lecture 05

McGill

# Outline

- Credit dataset involves 11 attributes. The objective is to model 'Balance'.
- Available attributes are: Income, Limit, Rating, Cards, Age, Education, Gender, Student, Married, and Ethnicity.
- Focus on quantitative variables: Income, Limit, Rating, Cards, Age, and Education.

# Notation $n$ observation, $p$ features

$$\boldsymbol{\varepsilon} = \begin{pmatrix} \varepsilon_1 \\ \vdots \\ \varepsilon_n \end{pmatrix}, \mathbf{y} = \begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix}, \mathbf{x} = \begin{pmatrix} x_1 \\ \vdots \\ x_p \end{pmatrix}, \boldsymbol{\beta} = \begin{pmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_p \end{pmatrix}$$

$$\mathbf{X}_{n \times p} = \begin{pmatrix} \mathbf{x}_1^\top \\ \vdots \\ \mathbf{x}_n^\top \end{pmatrix} = \begin{pmatrix} x_{11} & x_{12} & \dots & x_{1p} \\ x_{21} & x_{22} & \dots & x_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n1} & x_{n2} & \dots & x_{np} \end{pmatrix}$$

$$\mathbf{X}_{n \times (p+1)} = \begin{pmatrix} 1 & x_{11} & \dots & x_{1p} \\ 1 & x_{21} & \dots & x_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_{n1} & \dots & x_{np} \end{pmatrix}$$

# Matrix differentiation

$$
\begin{aligned}
\mathbf{y} &= \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\varepsilon} \\
S(\boldsymbol{\beta}) &= (\mathbf{y} - \mathbf{X}\boldsymbol{\beta})^\top (\mathbf{y} - \mathbf{X}\boldsymbol{\beta})
\end{aligned}
$$

What is the minimizer of $S(\boldsymbol{\beta})$?

$$\begin{aligned} \mathbf{y} &= \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\varepsilon} \\ S(\boldsymbol{\beta}) &= (\mathbf{y} - \mathbf{X}\boldsymbol{\beta})^\top (\mathbf{y} - \mathbf{X}\boldsymbol{\beta}) \end{aligned}$$

What is the minimizer of $S(\boldsymbol{\beta})$?

$$\frac{\partial \mathbf{X}\boldsymbol{\beta}}{\partial \boldsymbol{\beta}} = \mathbf{X}^\top$$

$$\frac{\partial \boldsymbol{\beta}^\top \mathbf{A}\boldsymbol{\beta}}{\partial \boldsymbol{\beta}} = (\mathbf{A} + \mathbf{A}^\top)\boldsymbol{\beta}$$

# Least squares

$$
\begin{aligned}
\min \quad & S(\boldsymbol{\beta}) \\
\frac{\partial S(\boldsymbol{\beta})}{\partial \boldsymbol{\beta}} &= 0 \\
(\mathbf{X}^\top \mathbf{X})\boldsymbol{\beta} &= \mathbf{X}^\top \mathbf{y} \\
\boldsymbol{\beta} &= (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{y}
\end{aligned}
$$

# Quantitative attributes



| | Balance | Income | Limit | Rating | Cards | Age | Education |
|---|---|---|---|---|---|---|---|
| **0** | 333 | 14.891 | 3606 | 283 | 2 | 34 | 11 |
| **1** | 903 | 106.025 | 6645 | 483 | 3 | 82 | 15 |
| **2** | 580 | 104.593 | 7075 | 514 | 4 | 71 | 11 |
| **3** | 964 | 148.924 | 9504 | 681 | 3 | 36 | 11 |
| **4** | 331 | 55.882 | 4897 | 357 | 2 | 68 | 16 |

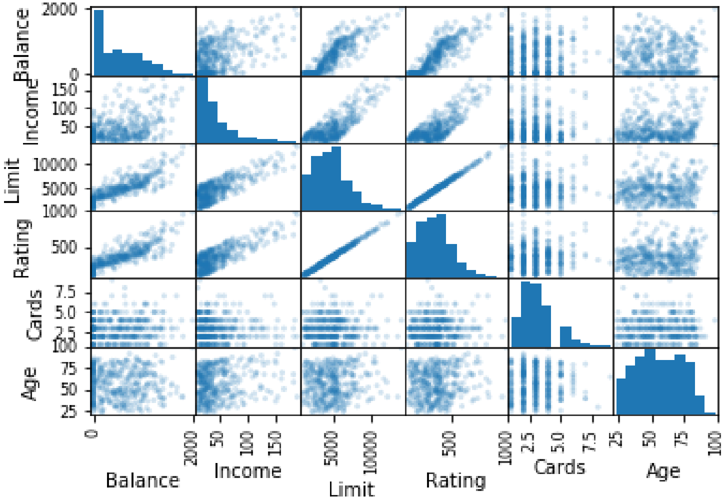
# Scatterplot Implementation



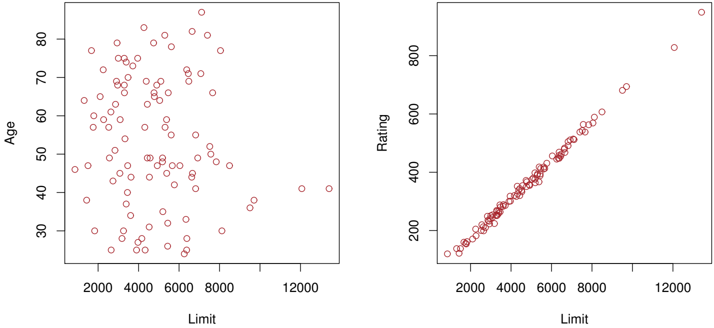
```
import pandas as pd
path='data/'
filename = path+'Credit.csv'
credit = pd.read_csv(filename)
credit = credit[['Balance','Income', 'Limit',
        'Rating', 'Cards', 'Age', 'Education']]
credit.head()
```

```
from pandas.plotting import scatter_matrix
%matplotlib inline
scatter_matrix(credit, alpha=0.2);
```

# Scatterplot

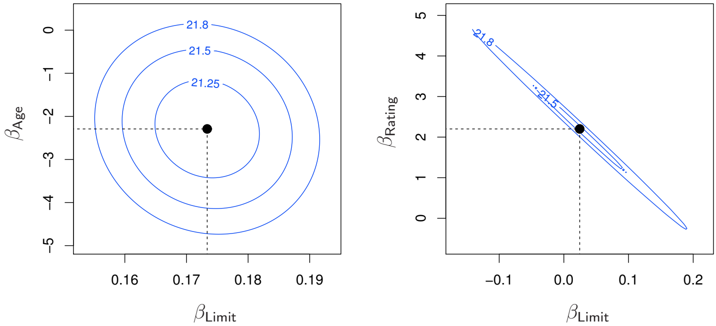


# Collinearity



# Effect of collinearity on estimation





# Effect of collinearity on estimation



| | | Coefficient | Std. error | t-statistic | p-value |
|---|---|---|---|---|---|
| Model 1 | `Intercept` | −173.411 | 43.828 | −3.957 | $< 0.0001$ |
| | `age` | −2.292 | 0.672 | −3.407 | 0.0007 |
| | `limit` | 0.173 | 0.005 | 34.496 | $< 0.0001$ |
| Model 2 | `Intercept` | −377.537 | 45.254 | −8.343 | $< 0.0001$ |
| | `rating` | 2.202 | 0.952 | 2.312 | 0.0213 |
| | `limit` | 0.025 | 0.064 | 0.384 | 0.7012 |

# All possible subsets

Suppose you have the response vector $\mathbf{y}$ and $\mathbf{x}_j, j = 1, \ldots p$ attributes.

- Fit $2^p$ possible models
- Choose the model with the best AIC or BIC.

# Forward

Suppose you have the response vector $\mathbf{y}$ and $\mathbf{x}_j, j = 1, \ldots p$ attributes.
Repeat the following algorithm until all attributes are in the model.

1. Initialize with the constant model $\mathbf{y} = \beta_0$
2. Compute $\hat{\mathbf{y}}$
3. Compute the residual $\mathbf{r} = \mathbf{y} - \hat{\mathbf{y}}$
4. Find $j'$ the most correlated attribute $\mathbf{x}_j$ with $\mathbf{r}$.
5. Enter $j'$ to the model
6. Go to 2

Select one of these models using AIC or BIC.

# Model size = 1

```
from sklearn.linear_model import LinearRegression
y = credit['Balance'].values
X = credit[['Income']].values
lr = LinearRegression()
lr.fit(X,y)
score1 = lr.score(X,y)
```

# Model size = 1

```
from sklearn.linear_model import LinearRegression
y = credit['Balance'].values
X = credit[['Income']].values
lr = LinearRegression()
lr.fit(X,y)
score1 = lr.score(X,y)

y = credit['Balance'].values
X = credit[['Limit']].values
lr = LinearRegression()
lr.fit(X,y)
score2 = lr.score(X,y)
```

# Model size = 1

```
from sklearn.linear_model import LinearRegression
y = credit['Balance'].values
X = credit[['Income']].values
lr = LinearRegression()
lr.fit(X,y)
score1 = lr.score(X,y)

y = credit['Balance'].values
X = credit[['Limit']].values
lr = LinearRegression()
lr.fit(X,y)
score2 = lr.score(X,y)

y = credit['Balance'].values
X = credit[['Rating']].values
lr = LinearRegression()
lr.fit(X,y)
score3 = lr.score(X,y)
```

Which attribute you choose?

# Model size = 2



```
y = credit['Balance'].values
X = credit[['Rating', 'Income']].values
lr = LinearRegression()
lr.fit(X,y)
score31 = lr.score(X,y)
```

# Model size $= 2$



```
y = credit['Balance'].values
X = credit[['Rating', 'Income']].values
lr = LinearRegression()
lr.fit(X,y)
score31 = lr.score(X,y)
```

```
y = credit['Balance'].values
X = credit[['Rating', 'Limit']].values
lr = LinearRegression()
lr.fit(X,y)
score32 = lr.score(X,y)
```

Which attribute you choose next?

# Credit data

| # Variables | Best subset | Forward stepwise |
|---|---|---|
| One | rating | rating |
| Two | rating, income | rating, income |
| Three | rating, income, student | rating, income, student |
| Four | cards, income student, limit | rating, income, student, limit |

# Backward

Suppose you have the response vector $\mathbf{y}$ and $\mathbf{x}_j, j = 1, \ldots p$ attributes.
Repeat the following algorithm until there is no attribute in the model.

1. Initialize with the full $\mathbf{y} = \beta_0 + \beta_1 \mathbf{x}_1 + \cdots \beta_p x_p$
2. Compute $\hat{\mathbf{y}}$
3. Compute the residual $\mathbf{r} = \mathbf{y} - \hat{\mathbf{y}}$ and $\mathrm{RSS} = \mathbf{r}^\top \mathbf{r}$
4. Find $j'$ the attribute the drops $\mathrm{RSS}$ the least possible
5. Remove $j'$ from the model
6. Go to 2

Select one of these models using AIC or BIC.

Least squares

$$\hat{\boldsymbol{\beta}} = \min(\mathbf{y} - \mathbf{X}\boldsymbol{\beta})^\top(\mathbf{y} - \mathbf{X}\boldsymbol{\beta})$$

Subset selection

$$\begin{aligned} \hat{\boldsymbol{\beta}} &= \min(\mathbf{y} - \mathbf{X}\boldsymbol{\beta})^\top(\mathbf{y} - \mathbf{X}\boldsymbol{\beta}) \\ \text{s.t.} &\quad \#\{\boldsymbol{\beta}_j \neq 0\} \leq s \end{aligned}$$

or

$$\begin{aligned} \hat{\boldsymbol{\beta}} &= \min(\mathbf{y} - \mathbf{X}\boldsymbol{\beta})^\top(\mathbf{y} - \mathbf{X}\boldsymbol{\beta}) \\ \text{s.t.} &\quad \sum_{j=1}^{p} I(\beta_j \neq 0) \leq s \end{aligned}$$

for a given model size $s$, where $I(.)$ is the indicator function.

# Towards shrinkage



for a given ball size $s$

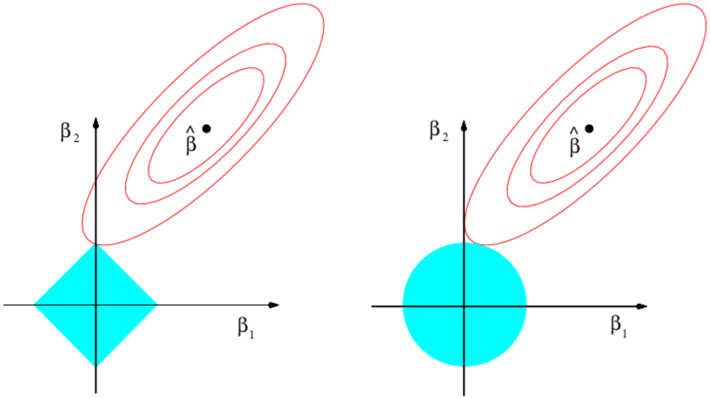
Ridge regression

$$\hat{\boldsymbol{\beta}} = \min(\mathbf{y} - \mathbf{X}\boldsymbol{\beta})^\top(\mathbf{y} - \mathbf{X}\boldsymbol{\beta})$$
$$\text{s.t.} \quad \sum_{j=1}^{p} \beta_j^2 \leq s$$

Lasso regression

$$\hat{\boldsymbol{\beta}} = \min(\mathbf{y} - \mathbf{X}\boldsymbol{\beta})^\top(\mathbf{y} - \mathbf{X}\boldsymbol{\beta})$$
$$\text{s.t.} \quad \sum_{j=1}^{p} |\beta_j| \leq s$$

# Shrinkage versus selection



For a given penalization constant $\lambda$
Ridge regression

$$\hat{\boldsymbol{\beta}} \;=\; \min(\mathbf{y}-\mathbf{X}\boldsymbol{\beta})^\top(\mathbf{y}-\mathbf{X}\boldsymbol{\beta}) + \lambda \sum_{j=1}^{p} \beta_j^2$$

Lasso regression

$$\hat{\boldsymbol{\beta}} \;=\; \min(\mathbf{y}-\mathbf{X}\boldsymbol{\beta})^\top(\mathbf{y}-\mathbf{X}\boldsymbol{\beta}) + \lambda \sum_{j=1}^{p} |\beta_j|$$

```
from sklearn.linear_model import Ridge
import numpy as np
y = credit['Balance'].values
X = credit[['Income', 'Limit', 'Rating',
        'Cards', 'Age', 'Education']].values
rr = Ridge(alpha=0, normalize=True)
rr.fit(X, y)
```

```
from sklearn.linear_model import Ridge
import numpy as np
y = credit['Balance'].values
X = credit[['Income', 'Limit', 'Rating',
        'Cards', 'Age', 'Education']].values
rr = Ridge(alpha=0, normalize=True)
rr.fit(X, y)


X_pred = np.array([15, 3000, 300, 2, 34, 16]).reshape(1,6)
rr.predict(X_pred)
```

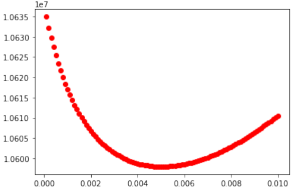
```
from sklearn.linear_model import Ridge
import numpy as np
y = credit['Balance'].values
X = credit[['Income', 'Limit', 'Rating',
        'Cards', 'Age', 'Education']].values
rr = Ridge(alpha=0, normalize=True)
rr.fit(X, y)

X_pred = np.array([15, 3000, 300, 2, 34, 16]).reshape(1,6)
rr.predict(X_pred)

rr = Ridge(alpha=10, normalize=True)
rr.fit(X, y)
rr.predict(X_pred)
```

# Ridge Generalized Cross-validation



```
from sklearn.linear_model import RidgeCV
alpha_values = np.linspace(0.0001, 0.01, num= 100)
rrcv = RidgeCV(alphas=alpha_values,
               normalize = True, store_cv_values = True)
rrcv.fit(X, y)
```

```
import matplotlib.pyplot as plt
cv_values = np.sum(rrcv.cv_values_, axis=0)
plt.plot(alpha_values, cv_values, 'or');
```



```
from sklearn.linear_model import Lasso
lr = Lasso(alpha = 0.1, normalize = True)
lr.fit(X,y)
lr.predict(X_pred)
```

# Lasso

```
from sklearn.linear_model import Lasso
lr = Lasso(alpha = 0.1, normalize = True)
lr.fit(X,y)
lr.predict(X_pred)
```

```
from sklearn.linear_model import LassoCV
lrcv = LassoCV(alphas = alpha_values, cv = 10, normalize = True)
lrcv.fit(X, y)
lrcv.alpha_
```

# Least Angle Regression

The least angle regression provides a fast framework similar to forward selection gives the solution path for lasso.

```
import matplotlib.pyplot as plt
from sklearn.linear_model import lars_path
alphas,_,coefs = lars_path(X, y, method='lasso', verbose=True)
alphas[1]
coefs[:,1]
```