MBAN 5110: PREDICTIVE MODELING

SESSION 4: REGULARIZATION METHODS

DR. ISIK BICER

Schulich School of Business | YORK UNIVERSITÉ UNIVERSITY

TODAY'S AGENDA

- Linear Regression: Review
- L1 regularization
- L2 regularization
- Tailored model

ANALYTICAL REPRESENTATION

Schulich School of Business York University

- y : Dependent variable (m observations)
- x : Independent variables
- ϵ : Error terms
- β : Coefficients

$$
\begin{gathered}
y_1 = \beta_0 + \beta_1 x_{11} + \beta_2 x_{12} + \beta_3 x_{13} + \cdots + \beta_n x_{1n} + \epsilon_1 \\
y_2 = \beta_0 + \beta_1 x_{21} + \beta_2 x_{22} + \beta_3 x_{23} + \cdots + \beta_n x_{2n} + \epsilon_2 \\
\vdots \\
y_m = \beta_0 + \beta_1 x_{m1} + \beta_2 x_{m2} + \beta_3 x_{m3} + \cdots + \beta_n x_{mn} + \epsilon_m
\end{gathered}
$$

MATRIX FORM

$$Y = \begin{vmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{vmatrix}; \quad \mathrm{B} = \begin{bmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_n \end{bmatrix}; \quad X = \begin{bmatrix} 1 & X_{11} & X_{12} & \cdots & X_{1n} \\ 1 & X_{21} & X_{22} & & X_{2n} \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 1 & X_{m1} & X_{m2} & \cdots & X_{mn} \end{bmatrix}; \mathrm{E} = \begin{vmatrix} \epsilon_1 \\ \epsilon_2 \\ \vdots \\ \epsilon_m \end{vmatrix}$$

- Transpose of a matrix:

$$\mathrm{E}^T = |\epsilon_1, \epsilon_2, \dots, \epsilon_m|$$

$$X^T = \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ X_{11} & X_{21} & X_{31} & & X_{m1} \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ X_{1n} & X_{2n} & X_{3n} & \cdots & X_{mn} \end{bmatrix}$$

ORDINARY LEAST SQUARES (OLS) ESTIMATION

$$Y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{bmatrix}; \quad \mathrm{B} = \begin{bmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_n \end{bmatrix}; \quad X = \begin{bmatrix} 1 & X_{11} & X_{12} & \cdots & X_{1n} \\ 1 & X_{21} & X_{22} & & X_{2n} \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 1 & X_{m1} & X_{m2} & \cdots & X_{mn} \end{bmatrix}; \mathrm{E} = \begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ \vdots \\ \epsilon_m \end{bmatrix}$$

- $Y = X\mathrm{B} + \mathrm{E}$
- Sum of square of errors:

$$\mathrm{E}^T\mathrm{E} = (Y - X\mathrm{B})^T(Y - X\mathrm{B}) = Y^TY - Y^TX\mathrm{B} - (X\mathrm{B})^TY + (X\mathrm{B})^TX\mathrm{B}$$

$$\mathrm{E}^T\mathrm{E} = Y^TY - 2Y^TX\mathrm{B} + \mathrm{B}^TX^TX\mathrm{B}$$

$$\frac{\partial \mathrm{E}^T\mathrm{E}}{\partial \mathrm{B}} = -2X^TY + 2X^TX\mathrm{B} = 0$$

$$\mathrm{B} = (X^TX)^{-1}X^TY$$

CONSISTENCY OF OLS ESTIMATION

- X and E should be independent
  - Error terms should be orthogonal to independent variables
  - In mathematical terms:

$$X^T\mathrm{E} = 0$$

$$X^T(Y - X\mathrm{B}) = 0$$

$$X^TY - X^TXB = 0$$

$$X^TY = X^TXB$$

$$(X^TX)^{-1}X^TY = (X^TX)^{-1}X^TXB$$

$$\mathrm{B} = (X^TX)^{-1}X^TY$$

- So, we reach the same result from the orthogonality condition.

PROBLEMS

- When there are too many independent variables:
  - For example, $x_1, x_2, x_3, \ldots, x_{50}, \ldots$
  - Some variables may be correlated to each other:
    - Inconsistent estimates
    - Coefficients may be close to zero

L1 REGULARIZATION: THE LASSO

THE LASSO

- Objective: Minimize the sum of squared residuals

$$(Y - X\mathrm{B})^T(Y - X\mathrm{B})$$

- But, we now add a constraint to limit the number of coefficients in the model:

$$\sum_{k=1}^{n} |\beta_k| \leq t$$

- We add the constraint as a penalty to the objective function

$$(Y - X\mathrm{B})^T(Y - X\mathrm{B}) + \alpha \sum_{k=1}^{n} |\beta_k|$$

THE LASSO

$$(Y - X\mathrm{B})^T (Y - X\mathrm{B}) + \alpha \sum_{k=1}^{n} |\beta_k|$$

- Let’s look at this function more carefully
- As the number of non-negative coefficients increases, so does the sum of squared residuals
- So, any attempt to minimize this function should jointly reduce the residuals and the number of coefficients

THE LASSO

$$(Y - XB)^T (Y - XB) + \alpha \sum_{k=1}^{n} |\beta_k|$$

- This formulation is quadratic
- Efficient solution is based on Least Angle Regression method
- In Python, the coefficients may change every time when the model is run

LEAST ANGLE REGRESSION

- Initialize $\beta_1, \beta_2, \beta_3, \ldots, \beta_n = 0$, and find the residual vector if these coefficients are used: $\boldsymbol{r_0}$
- Find the column that has the highest correlation with $\boldsymbol{r_0}$.
  - Suppose it is variable j. Increase β_j value towards its OLS estimate.
  - Recalculate the residual vector.
  - Stop at the point where variable j has no longer the highest correlation.
  - Repeat the steps with the new variable that has the highest correlation with $\boldsymbol{r_0}$

PYTHON PROGRAMMING

- Library: "sklearn"
  - Sublibrary: "linear_model"
  - Function: Lasso()
- Webpage for sklearn: https://scikit-learn.org
- Webpage for Lasso: https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.Lasso.html

PYTHON PROGRAMMING

- Generate dependent and independent variables!

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import statsmodels.api as sm
from sklearn import linear_model as lm
```

```
x_1 = np.random.uniform(10,40,100)
x_2 = np.random.uniform(-50,20,100)
x_3 = np.random.uniform(20,60,100)
x_4 = np.random.uniform(10,40,100)
x_5 = np.random.uniform(-50,20,100)
x_6 = np.random.uniform(20,60,100)
epsilon = np.random.normal(0,10,100)
```

```
y=-30+1.3*x_1+1.6*x_2+1.1*x_3+0.7*x_4-2.1*x_5-0.9*x_6+epsilon
```

PYTHON PROGRAMMING

- We first check the OLS results. To this end, we need to create the X matrix

```
X_ols=pd.DataFrame()
X_ols['Constant']=pd.Series(np.ones(100))
X_ols['X1'] = pd.Series(x_1)
X_ols['X2'] = pd.Series(x_2)
X_ols['X3'] = pd.Series(x_3)
X_ols['X4'] = pd.Series(x_4)
X_ols['X5'] = pd.Series(x_5)
X_ols['X6'] = pd.Series(x_6)
```

PYTHON PROGRAMMING

```
model_reg = sm.OLS(y,X_ols).fit()
model_reg.summary()
```

OLS Regression Results

Dep. Variable:	y	**R-squared:**	0.967
Model:	OLS	**Adj. R-squared:**	0.965
Method:	Least Squares	**F-statistic:**	454.0
Date:	Sun, 02 Oct 2022	**Prob (F-statistic):**	1.42e-66
Time:	14:07:07	**Log-Likelihood:**	-370.82
No. Observations:	100	**AIC:**	755.6
Df Residuals:	93	**BIC:**	773.9
Df Model:	6		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Constant	-29.3296	6.731	-4.357	0.000	-42.696	-15.963
X1	1.2386	0.118	10.526	0.000	1.005	1.472
X2	1.5405	0.053	29.295	0.000	1.436	1.645
X3	1.0986	0.090	12.203	0.000	0.920	1.277
X4	0.7359	0.118	6.229	0.000	0.501	0.971
X5	-2.1346	0.054	-39.287	0.000	-2.242	-2.027
X6	-0.9411	0.090	-10.503	0.000	-1.119	-0.763

PYTHON PROGRAMMING

- Let's generate some additional variables that are expected to be removed from the model by Lasso:

```
# Now we add some unrelated coefficients
x_7 = np.random.uniform(10,40,100)
x_8 = np.random.uniform(-50,20,100)
x_9 = np.random.uniform(20,60,100)
x_10 = np.random.uniform(10,40,100)
x_11 = np.random.uniform(-50,20,100)
x_12 = np.random.uniform(20,60,100)
```

PYTHON PROGRAMMING

```
# When we input the matrix, we don't need to add the column of ones because \
# Lasso automatically takes it into account

X_ext = X_ext.drop(columns=['Constant'])
model_lasso = lm.Lasso(alpha=1).fit(X_ext,y)
model_lasso.coef_
```

```
array([ 1.23952752,  1.53145407,  1.09882225,  0.69359863, -2.14047446,
       -0.92871513,  0.10059897, -0.01586398, -0.01655489, -0.03638864,
       -0.03742941,  0.05533014])
```

```
model_lasso = lm.Lasso(alpha=10).fit(X_ext,y)
model_lasso.coef_
```

```
array([ 1.10484424,  1.51390299,  1.01218205,  0.59247201, -2.09748408,
       -0.84828585,  0.        , -0.        , -0.        , -0.        ,
       -0.        ,  0.        ])
```

L2 REGULARIZATION: RIDGE REGRESSION

RIDGE REGRESSION

- https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.Ridge.html
- Objective: Minimize the sum of squared residuals

$$(Y - X\mathrm{B})^T(Y - X\mathrm{B})$$

- But, we now add a constraint to limit the number of coefficients in the model:

$$\mathrm{B}^T\mathrm{B} \leq t$$

RIDGE REGRESSION

$$(Y - X\mathrm{B})^T(Y - X\mathrm{B}) + \alpha \mathrm{B}^T\mathrm{B}$$

- The estimates for the Ridge regression can be obtained as an analytical expression:

$$\mathrm{B}_{ridge} = (X^T X + \alpha I)^{-1} X^T Y$$

- “Ridge()” function

```
model_ridge = lm.Ridge(alpha=10).fit(X_ext,y)
model_ridge.coef_
```

```
array([ 1.25127262,  1.53337585,  1.10937072,  0.70229948, -2.14587799,
       -0.93878662,  0.11997347, -0.01943572, -0.0270754 , -0.05503608,
       -0.04212088,  0.06266356])
```

```
model_ridge = lm.Ridge(alpha=10000).fit(X_ext,y)
model_ridge.coef_
```

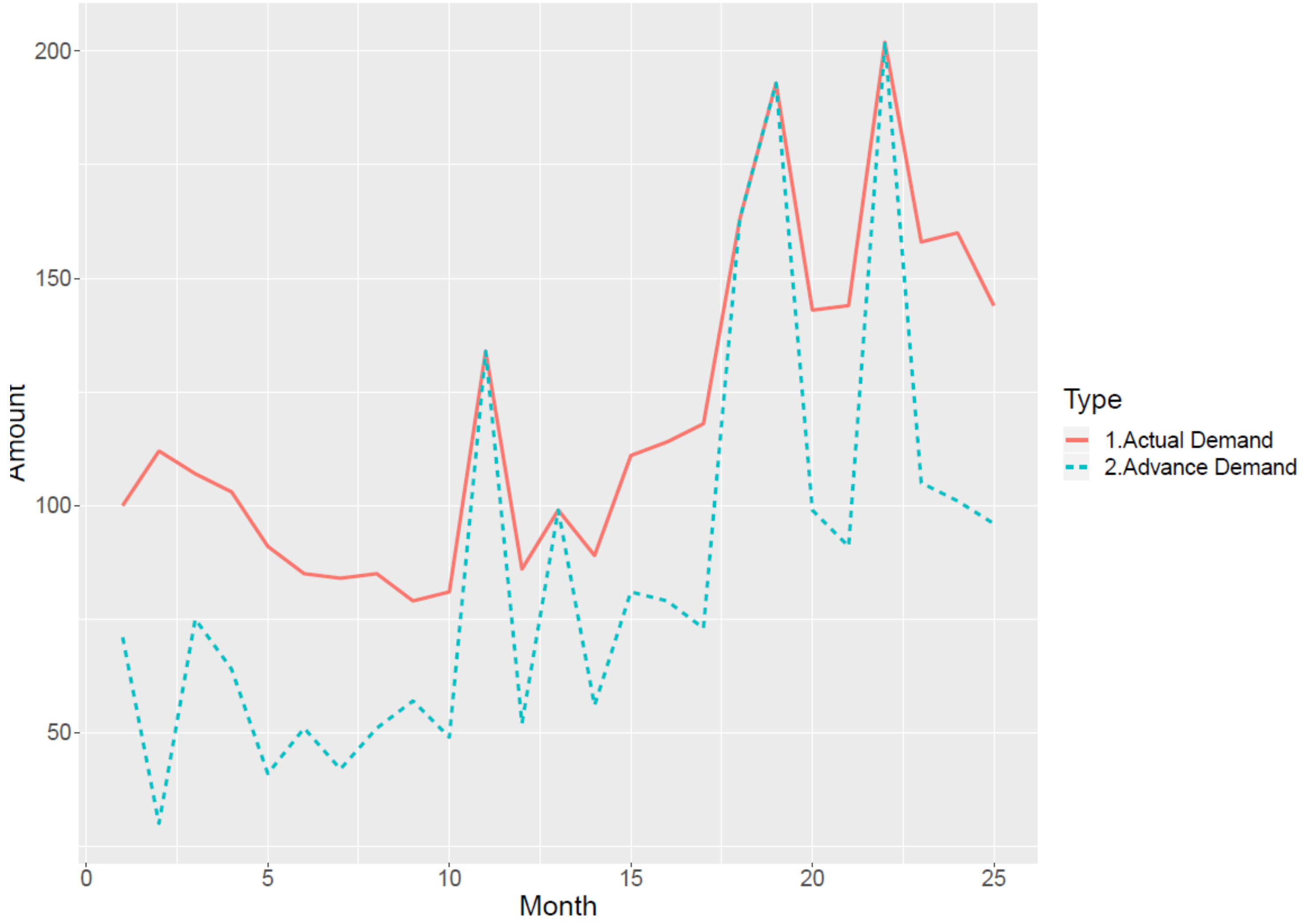
```
array([ 0.50321658,  1.21756962,  0.57935289,  0.32105205, -1.62079055,
       -0.42170529,  0.00375219, -0.07646285,  0.07164929,  0.05336775,
        0.01281994,  0.08499685])
```

- Remember: When alpha = 10, Lasso made most coefficients equal to zero

AN INTERESTING REGULARIZATION EXAMPLE

- Manufacturers collect orders from their customers in advance of the requested delivery date
- Time elapsed from the instant when an order is placed until its delivery is referred to as ***demand lead time***
- Demand lead time varies for each order between two weeks and several months
- The sum of the orders with a demand lead time that is longer than one month is referred to as ***advance demand***

AN INTERESTING REGULARIZATION EXAMPLE



AN INTERESTING REGULARIZATION EXAMPLE

- If advance demand is unexpectedly high, actual demand is equal to the advance demand (months 11, 13, 15, 18, 19 and 22)
- We also know that actual demand cannot be less than the advance demand
- Suppose we regress the demand values on the prior month's demand values (AR1: autoregressive model with one lag) and add the constraint the demand predictions cannot be less than advance demand

MODEL CONSTRUCTION

- Y: The vector of demand values from the second month to the last month (month 25)
- X: The vector of demand values from the first month to the twenty-fourth month
- L: The vector of advance demand values from the second month to the last month (month 25)

$$\xi^T \xi = (Y - X\beta)^T (Y - X\beta)$$

subject to: $X\beta \geq L$

ESTIMATION OF THE COEFFICIENTS

$$\beta_{REG} = (X^T X)^{-1}(X^T Y + 0.5X^T \lambda)$$

where $\lambda = 2(X(X^T X)^{-1} X^T)^{-1} L - 2Y$