On the benefits of regularization

Mohamed Ndaoud

INTRODUCTION

LINEAR MODELS are :

- **Powerful tools for making data-driven decisions** in industry (customer behavior, optimizing operations, assessing risk etc.).
- **Widely used** to drive growth and profitability.
- Transparent and **explainable**.

"Linear models are a **core component** *for statistical software that analyzes treatment effects. They are used in* **platforms** *where analysis is automated, as well as* **scientific studies** *where analysis is done locally and manually."*

$\Longrightarrow$ Content recommendations (take into account your past choices in movies, the types of genres you like, and what movies were watched by users that had similar tastes like yours).

[1]

1. Wong, J., Lewis, R., Wardrop, M. (2019). Efficient computation of linear model treatment effects in an experimentation platform. arXiv preprint arXiv :1910.01305.

LINEAR REGRESSION IN PYTHON

- Of course, data scientists don't typically calculate these β formula by hand ...
  $\longrightarrow$ Rely on Python and Sklearn package

In practice :

```
from sklearn.linear_model import LinearRegression
regressor = LinearRegression()
regressor.fit(X_train, y_train) # train the linear regression
regressor.coef_ # access to the coefficients
regressor.intercept_ # access to the intercept
```

UNDERSTANDING R2 : THE COEFFICIENT OF DETERMINATION

Another regression metric

R2 intuition :
Coefficient of Determination, measures how well the regression model fits the data.

R2 Definition :

$$R2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2} = 1 - \frac{\sum_{i=1}^{n}\epsilon^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2} = 1 - \frac{RSS}{\text{Total sum of squares}}$$

$$= 1 - \frac{\frac{1}{n}RSS}{\frac{1}{n}\sum_{i=1}^{n}(y_i - \bar{y})^2} = 1 - \frac{MSE}{var(y)}$$

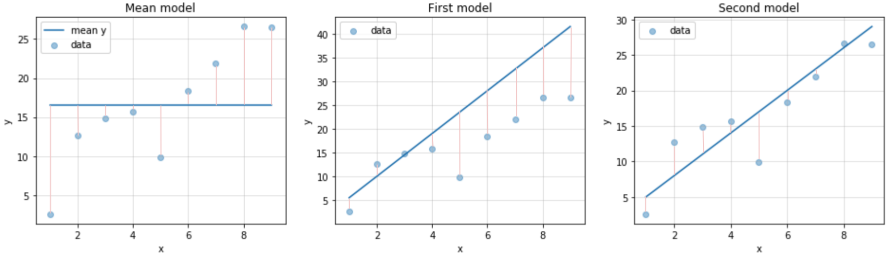
- **The residual sum of squares** measures the deviation of the model from the actual data.
- **The total sum of squares** measures the total variability of the dependent variable.
- **R2** is calculated as the ratio of the explained variation (by the model) to the total variation (outcome data).

UNDERSTANDING R2 : THE COEFFICIENT OF DETERMINATION

Another regression metric

Interpretation :

- **R2 close to 1 : Good fit**
  $R2 = 1 \longrightarrow \sum_i (y_i - \hat{y}_i)^2 = 0$
  The model predicts the values of the dependent variable perfectly.
- **R2 close to 0 : Poor fit**
  $R2 = 0 \longrightarrow \sum_i (y_i - \hat{y}_i)^2 = \sum_i (y_i - \bar{y})^2$
- **R2 $\leq$ 0 : Worse fit than a horizontal line**



Residual sum of squares : 489	Residual sum of squares : 749	Residual sum of squares : 106
Total sum of squares : 489	Total sum of squares : 489	Total sum of squares : 489
R2 = 1 - 489/489	R2 = 1 - 749/489	R2 = 1 - 106/489
R2 = 0	R2 = -0.53	R2 = 0.78

EASY INTERPRETATION OF LINEAR REGRESSION

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2$$

with x_1 being continuous and x_2 categorical.

- **Standardized coefficient** is measured in units of standard deviation.

  Example :

  A β_1 value of 2.25 indicates that a **change of one standard deviation** in the explicative variable results in a **2.25 standard deviations increase** in the target variable.

- **Categorical variable** :

  - Coefficient cannot be interpreted in the same way :

    $\longrightarrow$ It does not make sense to change x_2 by 1 standard deviation.

  - In general, coefficients are not meant to be interpreted individually

    $\longrightarrow$ But to be **compared to one another** in order to get a sense of the **importance of each variable** in the linear regression model.

EASY INTERPRETATION OF LINEAR REGRESSION - BOSTON HOUSE PRICES

KNN Regression with the Boston House Dataset [3]

INPUTS

1. CRIM - per capita crime rate by town
2. ZN - proportion of residential land zoned for lots over 25,000 sq.ft.
3. INDUS - proportion of non-retail business acres per town.
4. CHAS - Charles River dummy variable (1 if tract bounds river; 0 otherwise)
5. NOX - nitric oxides concentration (parts per 10 million)
6. RM - average number of rooms per dwelling
7. AGE - proportion of owner-occupied units built prior to 1940
8. DIS - weighted distances to five Boston employment centres
9. RAD - index of accessibility to radial highways
10. TAX - full-value property-tax rate per $10,000
11. PTRATIO - pupil-teacher ratio by town

[redacted]

13. LSTAT - % lower status of the population

OUTPUT

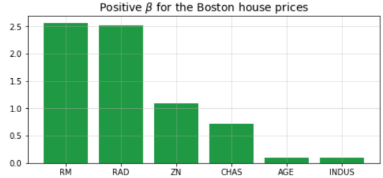
House Prices

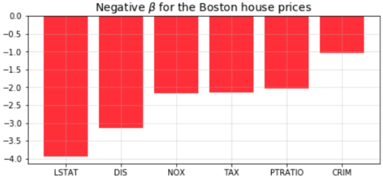
(expressed in $1000s)

3. The dataset for this project originates from the UCI Machine Learning Repository. The Boston housing data was collected in 1978 and each of the 506 entries represent aggregated data about 14 features for homes from various suburbs in Boston, Massachusetts.

EASY INTERPRETATION OF LINEAR REGRESSION - BOSTON HOUSE PRICES

$$\hat{y} = \hat{\beta}_0 + \hat{\beta}_{rm}x_{rm} + \hat{\beta}_{rad}x_{rad} + \hat{\beta}_{zn}x_{zn} + \hat{\beta}_{chas}x_{chas} + \hat{\beta}_{age}x_{age} + \hat{\beta}_{indus}x_{indus}$$
$$+ \hat{\beta}_{lstat}x_{lstat} + \hat{\beta}_{dis}x_{dis} + \hat{\beta}_{nox}x_{nox} + \hat{\beta}_{tax}x_{tax} + \hat{\beta}_{ptratio}x_{ptratio} + \hat{\beta}_{crim}x_{crim}$$





INPUTS

1. CRIM - per capita crime rate by town
2. ZN - proportion of residential land zoned for lots over 25,000 sq.ft.
3. INDUS - proportion of non-retail business acres per town.
4. CHAS - Charles River dummy variable (1 if tract bounds river; 0 otherwise)
5. NOX - nitric oxides concentration (parts per 10 million)
6. RM - average number of rooms per dwelling
7. AGE - proportion of owner-occupied units built prior to 1940
8. DIS - weighted distances to five Boston employment centres
9. RAD - index of accessibility to radial highways
10. TAX - full-value property-tax rate per $10,000
11. PTRATIO - pupil-teacher ratio by town

[redacted]

13. LSTAT - % lower status of the population

UPSIDES AND DOWNSIDES OF LINEAR REGRESSION

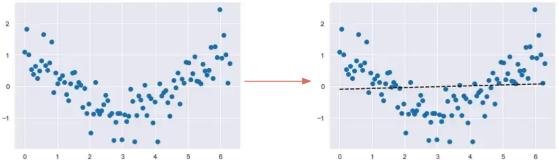
Upsides

- Easy to **understand** and interpret.

Downsides

- **Can not** effectively express **non-linear relationships**.

  - Example : The observations - with 1 feature - plotted reveal a **cosine curve pattern**.



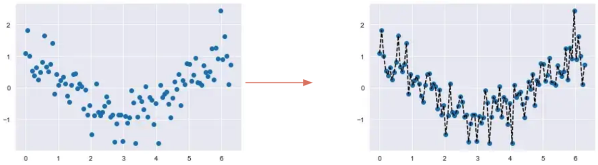
UPSIDES AND DOWNSIDES OF LINEAR REGRESSION

Downsides

- Very **prone to overfitting** when we have many features. $\Longrightarrow$ Complexity
  - Even more if some explicative variables are not useful predictors.

  - Example : Add **99** other features that contain **random values** (+ 1st previous feature)

  $\Longrightarrow$ No predictive influence to our target variable Y



UPSIDES AND DOWNSIDES OF LINEAR REGRESSION

Downsides

- Very **prone to overfitting** when we have many features. $\Longrightarrow$ Complexity

  - **Big data** : Common in real-world to collect a **lot of features** to be on the safe side.
  - **Feature engineering** : Create **even more features** with the initial feature.

$\Longrightarrow$ Easy to obtain **overfitting** in linear regression if they are useless as predictors.

UPSIDES AND DOWNSIDES OF LINEAR REGRESSION

What should we do to avoid overfitting in linear regression ?

REGULARIZATION TO AVOID OVERFITTING IN LINEAR REGRESSION

Regularization in machine learning : a process that changes the model to be "simpler" $\Longrightarrow$ Reduce complexity

Regularization in the context of linear regression : a process that **penalizes** β.

How to we penalize the coefficients ?

- **Add a penalty factor** to the RSS cost function.

$$RSS(\beta) = \sum_{i=1}^{N} (y_i - x_i^T \beta)^2$$

- The new cost function that needs to be minimized :

RSS + **a function with a penalty on coefficients**

What is the penalty function ?

REGULARIZATION TO AVOID OVERFITTING IN LINEAR REGRESSION

What is the penalty function ?

A **function** that depends on :

- β
- λ : Controls **how much we constraint the** β.
  Controls the **degree of fit** of the model on the training data
  $\Longrightarrow$ Ability to control overfitting

Different penalty functions : **L1 and L2**

- Change the **cost function**.
- Leads to **different fit behaviors**.

L1 REGULARIZATION TO AVOID OVERFITTING

- **L1 regularization** : penalizes the **absolute size** of the coefficients.

$$C(\beta) = \sum_{i=1}^{n} (y_i - x_i^T \beta)^2 + \lambda \sum_{j=1}^{k} |\beta_j|$$

- Results in the coefficients being **reduced to zero**.
- A higher λ leads to **more coefficients pushed to zero**.

What does it imply for a feature linked to a zero coefficient ?

- Does it remove that feature entirely ?
- Does it lead to a way of selecting features ?
- Does it complexify the model ?
- Can it take care of multicollinearity issues ?

L2 REGULARIZATION TO AVOID OVERFITTING

- **L2 regularization** : penalizes the **squared size** of the coefficients.

$$C(\beta) = \sum_{i=1}^{n}(y_i - x_i^T\beta)^2 + \lambda\sum_{j=1}^{k}\beta_j^2$$

- A higher λ leads to **more penalization**.
- Results in the coefficients being **shrunk**.

LASSO : REGRESSION ALGORITHM WITH L1 REGULARIZATION

- **LASSO**(**L**east **A**bsolute **S**hrinkage and **S**election **O**perator) : Regression analysis [4] that relies on the L1 penalty.

- Reminder :
  - Reduce the coefficients' sizes : they can get to 0 $\longrightarrow$ **feature selection**.
  - A higher λ leads to more coefficients pushed to zero $\longrightarrow$ the regression line is **underfitted**.

What type of model is it when λ is zero ?

4. In this course, we will focus on linear regression models using LASSO. But LASSO can also be applied to other regression models (example : generalized linear models, generalized estimating equations).

LASSO : REGRESSION ALGORITHM WITH L1 REGULARIZATION

- **LASSO**(**L**east **A**bsolute **S**hrinkage and **S**election **O**perator) : Regression analysis [5] that relies on the L1 penalty.

- Reminder :
  - Reduce the coefficients' sizes : they can get to 0 $\longrightarrow$ **feature selection**.
  - A higher λ leads to more coefficients pushed to zero $\longrightarrow$ the regression line is **underfitted**.

What type of model is it when λ is zero ?

- Applied on linear regression : $\lambda = 0 \longleftrightarrow$ linear regression model

5. In this course, we will focus on linear regression models using LASSO. But LASSO can also be applied to other regression models (example : generalized linear models, generalized estimating equations).

LASSO : REGRESSION ALGORITHM WITH L1 REGULARIZATION

- **LASSO** (**L**east **A**bsolute **S**hrinkage and **S**election **O**perator) : Regression analysis [6] that relies on the L1 penalty.

- **Remarks**

- The absolute function is **not differentiable**.
- There is **no closed formula** for the estimation β.
- Optimization strategy to find β : the coordinate descent algorithm.

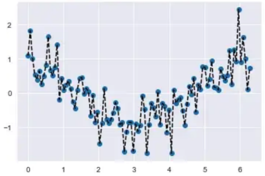
6. In this course, we will focus on linear regression models using LASSO. But LASSO can also be applied to other regression models (example : generalized linear models, generalized estimating equations).

LASSO WITH PYTHON

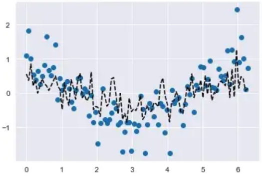
In practice :

```
from sklearn.linear_model import Lasso
lasso = Lasso(alpha=0.1)
lasso.fit(x_train, y_train)
lasso.coef_ # Acess to regularized beta
```

- α the penalty strength of the L1 regularization
- lasso.coef_ : $\hat{\beta}_{lasso}$



Linear Regression



Lasso

Homework : **Vary the parameter alpha and check how many coefficients were forced down to zero.**

RIDGE : REGRESSION ALGORITHM WITH L2 REGULARIZATION

- **Ridge** Regression analysis[7] that Relies on the **L2 penalty**.

- Reminder :
  - A higher λ leads to more coefficients approaching zero $\longrightarrow$ the regression line is **underfitted**.
  - Applied on linear regression : $\lambda = 0 \longleftrightarrow$ linear regression model

7. In this course, we will focus on linear regression models using LASSO. But LASSO can also be applied to other regression models (example : generalized linear models, generalized estimating equations).

RIDGE : REGRESSION ALGORITHM WITH L2 REGULARIZATION

- **Ridge** Regression analysis[8] that Relies on the **L2 penalty**.

- **Remarks**

- $\hat{\beta}_{ridge}$ always exists and is unique :

$$\begin{aligned}\hat{\beta}_{ridge} &= \arg\min_{\beta} \sum_{i=1}^{n} (Y - X\beta)^2 + \lambda \sum_{j=1}^{k} \beta_j^2 \\ &= \arg\min_{\beta} (Y - X\beta)^T (Y - X\beta) + \lambda ||\beta||_2^2\end{aligned}$$

- Solution

$$\boxed{\hat{\beta}_{ridge} = (X^T X + \lambda I^p)^{-1} X^T Y}$$

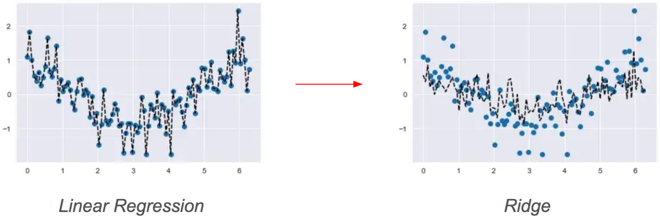
8. In this course, we will focus on linear regression models using LASSO. But LASSO can also be applied to other regression models (example : generalized linear models, generalized estimating equations).

RIDGE WITH PYTHON

In practice :

```
from sklearn.linear_model import Ridge
ridge = Ridge(alpha=0.1)
ridge.fit(x_train, y_train)
ridge.coef_ # Acess to penalized beta
```

- α the penalty strength of the L2 regularization
- ridge.coef_ : $\hat{\beta}_{ridge}$



Homework : Vary the parameter alpha
Are the coefficients higher when using Ridge instead of Lasso ?

LASSO OR RIDGE : DIFFERENT IMPACTS

Differences between L1 and L2 regularization in practice :

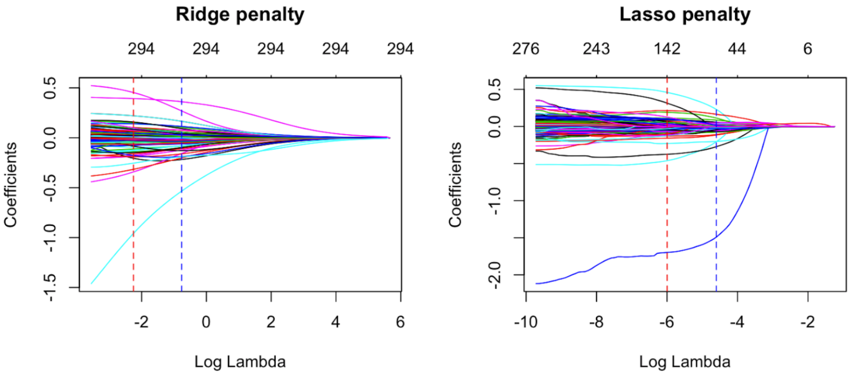
- **Lasso (L1 norm)** : Many coefficients being **pushed to zero**.
  - L1 norm is more **robust to outliers**
  - Unstable for different lambda values as the model changes significantly

- **Ridge (L2 norm)** : Most coefficients are **small but non-zero**.
  - L2 norm is better if you **want to consider outliers** when building your model (exponential increase in cost).
  - Ridge regression deals with multicollinearity [9]

- L1 has a **built-in feature selection** by pushing the coefficients to exactly 0, whereas L2 shrinks them to near 0 resulting in lower sparsity [10].

9. Investopedia : Multicollinearity is the occurrence of high intercorrelations among two or more independent variables in a multiple regression model.
10. Sparsity : A lot of zero elements.

LASSO OR RIDGE : DIFFERENT IMPACTS



A COMPROMISE BETWEEN LASSO AND RIDGE : ELASTIC-NET

Elastic-net

- Combines both Lasso and ridge penalties (L1 and L2)
- Usually results to better performance when tuned properly
- Benefits from both L1 and L2 regularization.
- More difficult to tune (2 parameters)
  - α_1 controls the L1 penalty ($\alpha_1 = 0$ ridge) :
  - α_2 controls the L2 penalty ($\alpha_2 = 0$ lasso) :

$$C(\beta) = \sum_{i=1}^{N}(y_i - x_i^T\beta)^2 + \alpha_1 \sum_{j=1}^{p}|\beta_j| + \alpha_2 \sum_{j=1}^{p}\beta_j^2$$
$$= (Y - \beta X)^T(Y - \beta X) + \alpha_1||\beta||_1 + \alpha_2||\beta||_2^2$$

A COMPROMISE BETWEEN LASSO AND RIDGE : ELASTIC-NET

- Alternatively, instead of using two parameters α_0 and α_1, we can use (sklearn formula) :
  - *alpha*
  - $L1_{ratio}$

$$L1_{ratio} = 0 \longrightarrow \text{ridge regression}$$
$$L1_{ratio} = 1 \longrightarrow \text{Lasso regression}$$

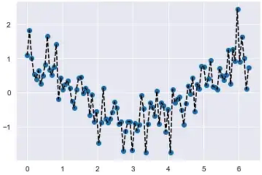
$$\begin{aligned} C(\beta) &= \sum_{i=1}^{N} (y_i - x_i^T \beta)^2 + \alpha L1_{ratio} \sum_{j=1}^{p} |\beta_j| + \alpha(1 - L1_{ratio}) \sum_{j=1}^{p} \beta_j^2 \\ &= (Y - \beta X)^T (Y - \beta X) + \alpha L1_{ratio} ||\beta||_1 + \alpha(1 - L1_{ratio}) ||\beta||_2^2 \end{aligned}$$

ELASTIC-NET IN PYTHON

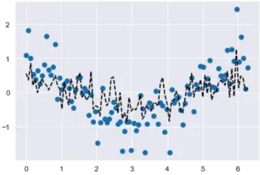
In practice :

```
from sklearn.linear_model import ElasticNet
elnet = ElasticNet(alpha=0.5, l1_ratio=0.5)
elnet.fit(x_train, y_train)
elnet.coef_ # Acess to penalized beta
```

- $L1_{ratio}$ together with the α should be tuned
- elnet.coef_ : $\hat{\beta}_{elnet}$



Linear Regression



Ridge

DATA SCALING

Why data scaling transformations are important ?

- Some models **won't work properly** if the features have different orders of magnitude. (Reminder : KNN)

- What about linear regression ?
  - Without scaling transformations : **Difficulty of comparison**.
  - Coefficients without scaling transformations should not be used to drop or rank predictors.

- Application in the code :
  - First transform the data before feeding it into the model (if mandatory) !
  - Data transformation is **calibrated on the train data set**, and not the test set !

DATA SCALING

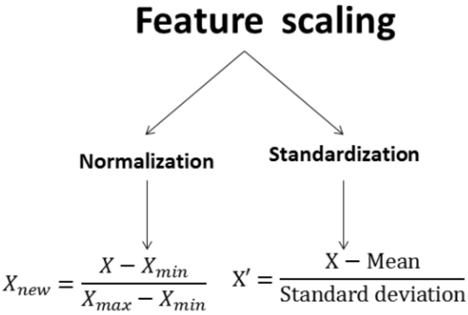
What are the main scaling transformations ?

Normalization

- Every feature is scaled into a 0-1 interval.
- Good if small number of outliers.

Standardization

- Mean $= 0$
- Variance $= 1$
  $\longrightarrow$ magnitude $= 1$



[a]

[a] Source : https ://www.naukri.com/learning/articles/normalization-and-standardization/

DATA SCALING WITH PYTHON

Normalization

```
import numpy as np
from sklearn.preprocessing import MinMaxScaler
data = np.array([[100, 0.001], [8, 0.05], [50, 0.005], [88, 0.07], [4, 0.1]])
scaler = MinMaxScaler() # define min max scaler
scaled = scaler.fit_transform(data) # transform data
print(scaled)
```

Standardization

```
import numpy as np
from sklearn.preprocessing import StandardScaler
x = np.array([[100000, 150000, 350000, 200000], [1, 2, 3, 2]])
x = x.T  convert rows to columns
print("Before standardization:", x)
scaler = StandardScaler()
x_standardized = scaler.fit_transform(x)
print("After standardized:", x_standardized)
# Standardizing manually:
x_std_manual = (x - np.mean(x, axis=0)) / np.std(x, axis=0)
print("After standardized manually:", x_std_manual)
```