

Machine Learning and Programming in Python

Lecture for Master and PhD students

Chair of Data Science in Economics

Ruhr University Bochum

Summer semester 2024

Lecture 5

Linear Regression

Linear Regression

- Linear regression is a simple approach to **Supervised Learning**. It assumes that the dependence of Y on X_1, X_2, \dots, X_p is linear
- Although it may seem overly simplistic, linear regression is extremely useful both conceptually and practically
- Starting point for many more advanced methods in Data Science
- If the linear functional relationship is not the true one between X and y , then the performance/prediction accuracy will be bad

Questions that might be asked:

- Is there a relationship between y und X ?
- How strong is the relationship between y und X ?
- Which variables in X contribute to y ?
- How well can the strength of the relationship between a variable in X and y be estimated?
- What will be our prediction for y ? How accurately can we predict y ?
- Is the relationship between X and y (the f) linear?
- Is there synergy among the variables in X (interaction terms)?

- We assume a model

$$Y = \beta_0 + \beta_1 * X + \epsilon$$

where β_0 and β_1 are two unknown constants that represent the intercept and slope, also known as coefficients or parameters, and ϵ is the error term.

- Given some estimates $\hat{\beta}_0$ and $\hat{\beta}_1$ for the model coefficients, we predict y using

$$\hat{y} = \hat{\beta}_0 + \hat{\beta}_1 * x$$

where \hat{y} indicates a prediction of Y on the basis of $X = x$. The hat symbol denotes an estimated value

- Let $\hat{y}_i = \hat{\beta}_0 + \hat{\beta}_1 * x_i$ be the prediction for Y based on the i th value of X.

Then $e_i = y_i - \hat{y}_i$ represents the i th residual

- We define the residual sum of squares (RSS) as

$$\text{RSS} = e_1^2 + e_2^2 + \dots + e_n^2,$$

or equivalently as

$$\text{RSS} = (y_1 - \hat{\beta}_0 - \hat{\beta}_1 * x_1)^2 + (y_2 - \hat{\beta}_0 - \hat{\beta}_1 * x_2)^2 + \dots + (y_n - \hat{\beta}_0 - \hat{\beta}_1 * x_n)^2$$

- The least squares approach chooses $\hat{\beta}_0$ and $\hat{\beta}_1$ to minimize the RSS.

The minimizing values can be shown to be

$$\hat{\beta}_1 = \frac{\sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^n (x_i - \bar{x})^2}$$

$$\hat{\beta}_0 = \bar{y} - \hat{\beta}_1 \bar{x}$$

where $\bar{y} = \frac{1}{n} \sum_{i=1}^n y_i$ and $\bar{x} = \frac{1}{n} \sum_{i=1}^n x_i$ are the sample means.

- The OLS (ordinary least squares) estimator chooses the β s (for p X-variables and the intercept), in order to minimise the RSS:

$$\begin{aligned}\min \text{RSS} &= \sum_{i=1}^n (y_i - \hat{y}_i)^2 \\ &= \sum_{i=1}^n (y_i - \hat{\beta}_0 - \hat{\beta}_1 * x_{i1} - \hat{\beta}_2 * x_{i2} - \dots - \hat{\beta}_p * x_{ip})^2\end{aligned}$$

- The OLS estimator works well, when $n \gg p$ and when Y is approximatively linearly related to X:
 - ▶ Small Bias and small Variance
 - ▶ Easy to interpret
 - ▶ Usable for inference

Regularisation

Regularisation

- If n only a bit larger than p , OLS models have high variance, overfitting and making poor predictions.
- If $n \leq p$, no unique solution. (see explanations during the lecture)
- Regularisation: Systematic shrinkage of OLS coefficients:
 - ▶ Can get a big reduction in variance at the cost of some bias.
 - ▶ Improves OLS, improves the prediction
 - ▶ Like imposing a cost on complexity to reduce overfitting.
 - ▶ For example: Ridge (L2 regularisation), Lasso (L1 regularisation).
- What are these methods and why do they work?

Ridge Regression

- Consider the following example:

Y , X and $f(X)$

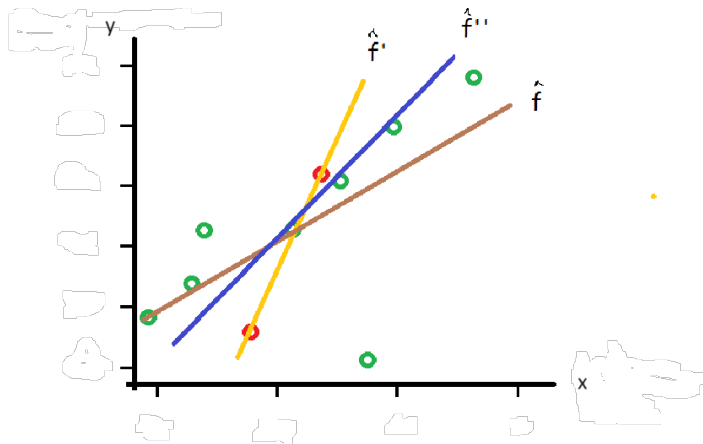
Whole sample: green and red dots

True relationship $f(X)$ is the brown-coloured line

Small training data sample: red dots

OLS fit from training data sample is the yellow-coloured curve

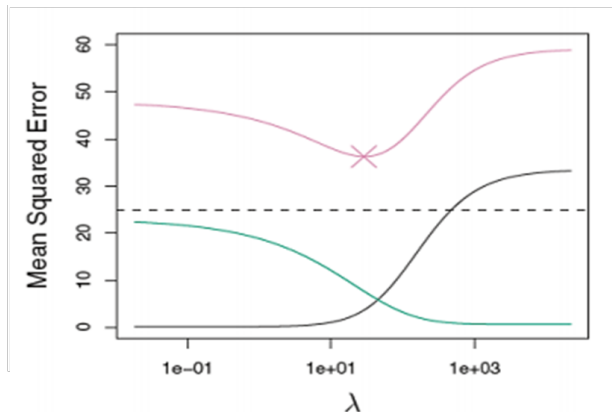
Ridge regression fit: blue line



- OLS: Estimate β s in order to minimise:
$$RSS = \sum_{i=1}^n (y_i - \hat{y}_i)^2$$
- Ridge: Estimate β s in order to minimise:
$$RSS + \lambda \sum_{j=1}^p \beta_j^2$$
- $\lambda \sum_{j=1}^p \beta_j^2$ is the penalty term

- λ is a model tuning parameter, cost of complexity
- $\lambda \sum_{j=1}^p \beta_j^2$ shrinks β s to zero (only the slope coefficients but not the intercept β_0)
- $\lambda = 0$ yields the standard OLS solution
- $\lambda \Rightarrow \infty$ makes β s approach 0 asymptotically
 - ▶ y will be less sensitive to changes of X, the larger gets λ
 - ▶ Ridge penalizes large $|\beta|$
- different λ imply different sets of $\hat{\beta}$

- Ridge regression on simulated data for $n=50$, $p=45$
- Squared bias (black), variance (green), test MSE (purple).
- For $\lambda = 0$, no bias but high variance.
- As λ rises, variance falls fast with small increase in bias
- Aim: to minimise the MSE



- OLS coefficients are scale **equivariant**
 - If X_j scales by a constant c , $\hat{\beta}_j$ scales by $\frac{1}{c}$.
 - E.g. if X_j is weight in kg, and is rescaled to pounds, the original $\hat{\beta}_j$ is divided by 2.2.
 - $\hat{\beta}_j X_j$ remains constant.
- Ridge regression coefficients are **not** scale equivariant
 - $\hat{\beta}_j^R X_j$ depends on λ and the scale of X_j .
- Solution: standardise the predictors:


$$\tilde{X}_j = \frac{X_j}{\sigma_{X_j}}$$

- All X s are in the same scale (1 unit change = 1 standard deviation)
- Then Ridge regression fit does not depend on scale of X .

Lasso Regression

- The Lasso is an alternative to Ridge regression.
- It picks β s to minimise $\sum_{i=1}^n (y_i - \hat{y}_i)^2 + \lambda \sum_{j=1}^p |\beta_j|$
- The Lasso also shrinks the β s towards zero.
- But unlike the Ridge regression, the Lasso forces some β s to be exactly 0 for large enough λ .
- Lasso yields sparser models - fewer coefficients - easier to interpret.

- The Ridge regression solves the following problem:

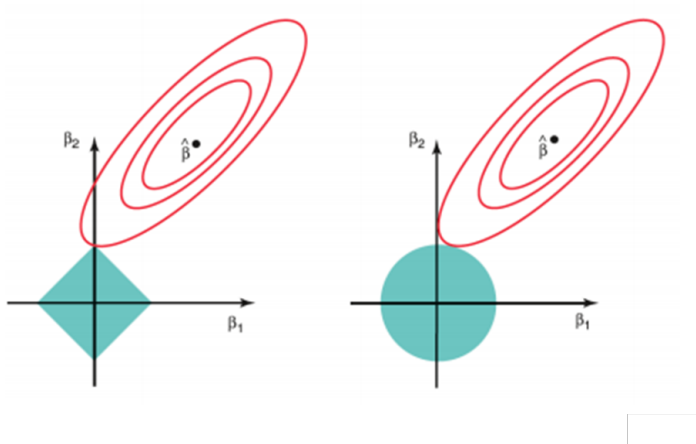

$$\min_{\beta} \sum_{i=1}^n (y_i - \hat{y}_i)^2 \text{ subject to } \sum_{j=1}^p \beta_j^2 \leq c$$

- The Lasso solves:

$$\min_{\beta} \sum_{i=1}^n (y_i - \hat{y}_i)^2 \text{ subject to } \sum_{j=1}^p |\beta_j| \leq c$$

In the following graph:

- the ellipses are $\hat{\beta}$ values that form “iso-RSS” contours
- the ellipses are centered on the standard OLS minimum of $\hat{\beta}$ s and RSS increases from inwards to outwards
- the shaded diamond and circle are the Lasso and Ridge constraints
- the Lasso constraint has corners on the axes: The ellipse will sooner intersect at a corner (than for Ridge), hence yield a zero coefficient



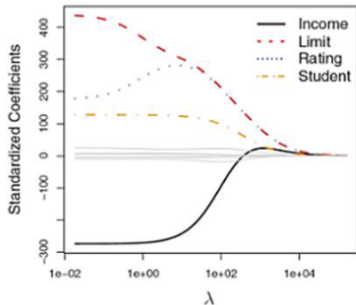
Example: Data set with several X-variables to explain credit default

The following Figures show the β coefficients

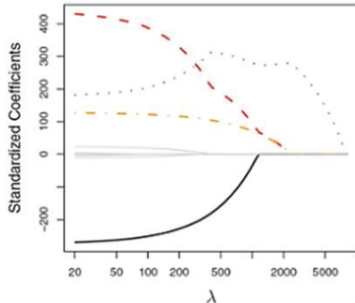
We have the OLS coefficients when $\lambda = 0$

Change of Ridge and Lasso coefficients, when tuning parameter $\lambda = 0$ changes

Ridge



Lasso



Elastic net

- Elastic net is based both on Ridge and Lasso regression
- Estimate β s in order to minimise:
$$\sum_{i=1}^n (y_i - \hat{y}_i)^2 + \lambda_1 \sum_{j=1}^p \beta_j^2 + \lambda_2 \sum_{j=1}^p |\beta_j|$$
- If $\lambda_2 = 0$, then we have Ridge regression. If $\lambda_1 = 0$ we have Lasso
- May use cross-validation to determine the best ratio between L1 and L2 penalty strength
- It is important to first standardize our data before feeding it into our regularised model

Summary

- The Lasso:
 - Yields a simpler model.
 - Will perform better if only a subset of the X have non-trivial coefficients; forces the remaining X with small coefficients to be zero.
- Ridge regression does better if most or all of the X matter and have roughly similar-sized coefficients.
- But...ex ante often not known how Y is related to different predictors; use cross-validation to pick the best model.
- Elastic net uses both the L1 and L2 penalty

Scikit-learn in Python

Scikit-learn in Python

- So far we focussed on processing data \Rightarrow use Numpy and Pandas to handle and import data
- Scikit-Learn provides a range of Supervised and Unsupervised Learning algorithms

<https://scikit-learn.org/stable/>

- Comes with some data sets for learning (Iris, Digits, formerly: Boston house prices, etc.)

https://scikit-learn.org/stable/datasets/toy_dataset.html

https://scikit-learn.org/stable/datasets/real_world.html

Scikit-learn facilitates:

- Data processing and normalisation
- Creating training and test data samples
- Cross-validation
- Model evaluation

How to implement a ML model in scikit-learn in Python

- 1. Import the model that one intends to use
 - ▶ All scikit-learn ML models are implemented as Python classes
 - ▶ For example:

```
from sklearn.linear_model import LinearRegression
```

- ▶ https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LinearRegression

- 2. Set up the specific model - instantiate the class

- ▶ For example:

```
model1 = LinearRegression()
```

- 3. Fit the model to the training data and store the parameter estimates
 - ▶ The data will include the X variables and the Y (outcome or classes)
 - ▶ For example:

```
model1.fit(X_train, Y_train)
```

- 4. Predict Y for the test data

- ▶ For example:

```
predictions1 = model1.predict(X_test)
```

Data Project

Data Project

- 1. Beer consumption, worldwide
- <https://worldpopulationreview.com/country-rankings/beer-consumption-by-country>
- Use Numpy and Pandas in Python
 - ▶ analyse the data set

- 2. Boston house prices
- formerly built-in data set in scikit-learn
- Use scikit-learn in Python
 - ▶ analyse the data set
 - ▶ estimate Linear regression, Lasso and Ridge regression

Literature

Literature:

James, Witten, Hastie, Tibshirani, Taylor (2023), An Introduction to Statistical Learning, Springer, Chapter 3, pp. 69 - 99; Chapter 6, pp. 240-253.