Week 2 - Linear Regression (Part 2)

-- **On our last episode**

We observe a certain quantity (e.g. the **effective life** of a certain industrial tool before it needs maintenance) and we want to make a model for it based on **how much we stress the tool** when using it (e.g. the rotation speed of its blade) and **its brand** (one of two options: A and B).

- $Y = \texttt{life}$
- $X_1 = \texttt{rpm}$
- $X_2 = \texttt{brand}$

We collect data from different tools on different factories, so to obtain a dataset like

$$\mathcal{D} = \{(X_{i1}, X_{i2}, Y_i)\}_{i \leq n}$$

and our linear model says that

$$Y_i = \beta_0 + \beta_1 X_{i1} + \beta_2 X_{i2} + \varepsilon_i$$

Our goal is to estimate the parameters **leveraging** the fact of having several examples. For this, we define the estimator as:

$$\hat{\beta} = \min_{\beta \in \mathbb{R}^3} \sum_{i=1}^{n} \Big(Y_i - (\beta_0 + \beta_1 X_{i1} + \beta_2 X_{i2}) \Big)^2$$

which can be reorganized as

$$\boldsymbol{X} = \text{design matrix} = \begin{bmatrix} 1 & X_{11} & X_{12} \\ \vdots & \vdots & \vdots \\ 1 & X_{n1} & X_{n2} \end{bmatrix} \in \mathbb{R}^{n \times 3} \quad \text{and} \quad \boldsymbol{y} = \begin{bmatrix} Y_1 \\ \vdots \\ Y_n \end{bmatrix} \in \mathbb{R}^n$$

and so the linear model is written as

$$\boldsymbol{y} = \boldsymbol{X}\beta + \varepsilon \quad \text{with} \quad \beta = \begin{bmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \end{bmatrix} \in \mathbb{R}^3 \quad \text{and} \quad \varepsilon \sim \mathcal{N}(0, \sigma^2 \boldsymbol{I}_n) \in \mathbb{R}^n$$

and we have that

$$\hat{\beta} = \min_{\beta \in \mathbb{R}^3} \|\boldsymbol{X}\beta - \boldsymbol{y}\|^2 = (\boldsymbol{X}^T \boldsymbol{X})^{-1} \boldsymbol{X}^T \boldsymbol{y}$$

-- **Regression with categorical variables**

Note that this example has a very important particularity that we should know how to handle: X_2 is a **categorical variable**, i.e. $X_2 \in \{A, B\}$

The standard way of handling this is to use **one-hot encoding** and represent X_2 as a binary vector:

$$X_2 = \begin{cases} 0 & \text{if tool is from brand A} \\ 1 & \text{if tool is from brand B} \end{cases}$$

In this case, when the tool is from brand A, our linear model simplifies to

$$Y = \beta_0 + \beta_1 X_1 + \varepsilon$$

whereas for brand B we have

$$Y = (\beta_0 + \beta_2) + \beta_1 X_1 + \varepsilon$$

showing that β_2 expresses a measure of the difference in mean tool life expectancy resulting from changing from brand A to brand B.

The example is from Montgomery et al. "Introduction to linear regression analysis" and it gives a table of values like this

```
> df
    life  rpm brand
1  18.73  610     A
2  14.52  950     A
3  17.43  720     A
4  14.54  840     A
5  13.44  980     A
6  24.39  530     A
7  13.34  680     A
8  22.71  540     A
9  12.68  890     A
10 19.32  730     A
11 30.16  670     B
12 27.09  770     B
13 25.40  880     B
14 26.05 1000     B
15 33.49  760     B
16 35.62  590     B
17 26.07  910     B
18 36.78  650     B
19 34.95  810     B
20 43.67  500     B
```

If we run `lm(life ~ rpm + brand)` in R we get the following result:

```
Call:
lm(formula = life ~ rpm + brand, data = df)

Residuals:
    Min      1Q  Median      3Q     Max
-5.5527 -1.7868 -0.0016  1.8395  4.9838

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 36.98560    3.51038  10.536 7.16e-09 ***
rpm         -0.02661    0.00452  -5.887 1.79e-05 ***
brandB      15.00425    1.35967  11.035 3.59e-09 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 3.039 on 17 degrees of freedom
Multiple R-squared:  0.9003, Adjusted R-squared:  0.8886
F-statistic: 76.75 on 2 and 17 DF,  p-value: 3.086e-09
```

Where we see that the `brand` predictor has become `brandB` because of the one-hot encoding done by R

The summary of `lm` gives us:

- Parameters β_0, β_1, and β_2 are not all equal to zero (F-test rejects null hypothesis)
- Parameter β_0 is different than zero (t-test rejects null hypothesis) $\Rightarrow$ the intercept is not zero
- Parameter β_1 is different than zero (t-test rejects null hypothesis) $\Rightarrow$ the rotation speed of the blade has an influence over the life expectancy for the cutting tool. Since $\beta_1 < 0$, the faster the blade turns the shorter the life of the tool.
- Parameter β_2 is different than zero (t-test rejects null hypothesis) $\Rightarrow$ the tool's brand has an effect over the life expectancy. Since $\beta_2 > 0$, tools from brand B have in average longer life expectancy.

We have only considered the simplest case with categorical variables: one categorial variable with only two levels.

There are of course much more complex settings.

1. For instance, we could have considered an **extra brand**, i.e. $\texttt{brand} \in \{A, B, C\}$ (multi-level analysis).

In this case, the one-hot encoding for variable X_2 would require two extra variables $X_2^{(b)}$ and $X_2^{(c)}$ so that

$$X_2^{(b)} = \begin{cases} 0 & \text{if tool is from brand A} \\ 1 & \text{if tool is from brand B} \end{cases} \quad \text{and} \quad X_2^{(c)} = \begin{cases} 0 & \text{if tool is from brand A} \\ 1 & \text{if tool is from brand C} \end{cases}$$

and the full linear model becomes

$$Y = \beta_0 + \beta_1 X_1 + \beta_2^{(b)} X_2^{(b)} + \beta_2^{(c)} X_2^{(c)} + \varepsilon_i$$

2. We could have included **more categorical variables**.

-- **Plan for today:**

- Define what makes a model good
- Estimating the quality of the model
- Comparing and selecting models

-- **What makes a model good?**

Consider the problem of **fitting a polynomial to a set of data points** that looks like this:

> **Figure 1**

We will fit a polynomial model to describe the dataset as follows:

$$Y = \beta_0 + \sum_{j=1}^{p} \beta_j X^j + \varepsilon \quad \text{with} \quad \varepsilon \sim \mathcal{N}(0, \sigma^2)$$

Although the features are non-linear in terms of the variable X, we can see this model as multiple linear regression in which $X_1 = X, X_2 = X^2, \ldots, X_p = X^p$ and, therefore,

$$Y = \beta_0 + \sum_{j=1}^{p} \beta_j X_j + \varepsilon \quad \text{with} \quad \varepsilon \sim \mathcal{N}(0, \sigma^2)$$

- We will do a first try with $p = 3$

Using $\mathbb{R}$ we can write: `summary(lm(yy ~ xx + I(xx^2) + I(xx^3)))` where function `I` ensures that the power operation happens over `xx` before it is used in the formula of the linear regression. We get:

```
Call:
lm(formula = yy ~ xx + I(xx^2) + I(xx^3))

Residuals:
    Min      1Q  Median      3Q     Max
-61.339 -12.227   0.612  13.944  48.409

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  -4.6731     3.1008  -1.507   0.1351
xx            2.5517     1.7729   1.439   0.1533
I(xx^2)      -0.6901     0.2719  -2.538   0.0128 *
I(xx^3)       0.9374     0.1062   8.826 4.93e-14 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 20.67 on 96 degrees of freedom
Multiple R-squared:  0.8717,    Adjusted R-squared:  0.8677
F-statistic: 217.4 on 3 and 96 DF,  p-value: < 2.2e-16
```

Plotting the approximation we get. Looks good, right?

> **Figure 2**

- Consider now that $p = 20$. The resulting curve that we get is:

> **Figure 3**

The new curve is more **wiggly** and one could say that it fits more closely the dataset...

Which option is best? How to decide the most appropriate order of the polynomial?

-- **Estimating the quality of a model**

Suppose we have some phenomenon that we want to study with a model and we assume that it can be described as

$$y = f(x) + \varepsilon \quad \text{with} \quad \mathbb{E}[\varepsilon] = 0 \quad \text{and} \quad \text{Var}(\varepsilon) = \sigma^2$$

Given a dataset $D = \{(x_i, y_i)\}_{i \leq n}$ where each $(x_i, y_i) \sim p(x, y)$ we can estimate an approximation for f denoted by $\hat{f}_D$, so that for any x our estimate for the corresponding y is $\hat{y}_D = \hat{f}_D(x)$.

For example, in multiple linear regression we have $\hat{y}_D = x^T \hat{\beta}_D$

1. A first approach of saying whether a model is a good one would be to evaluate whether for a given (x, y) we have a small value for

$$\left(y - \hat{f}_D(x)\right)^2$$

2. However, in fact we would appreciate if this quantity was **small for any choice of x and y** so our goal

should rather be that of having a model that has a small value of

$$\mathbb{E}_{(x,y)\sim p(x,y)}\left[\left(y-\hat{f}_D(x)\right)^2\right]=\iint\left(y-\hat{f}_D(x)\right)^2 p(x,y)\,\mathrm{d}x\,\mathrm{d}y$$

3. We should also note that $\hat{f}_D$ depends on how the points of the dataset D were sampled, so the most appropriate quantity to assess the quality of our model should take the average of all possible ways in which the dataset can be obtained by

$$\mathrm{MSE}=\mathbb{E}_{D\sim p^n(x,y)}\,\mathbb{E}_{(x,y)\sim p(x,y)}\left[\left(y-\hat{f}_D(x)\right)^2\right]=\iiint\left(y-\hat{f}_D(x)\right)^2 p(x,y)\,p(D)\,\mathrm{d}x\,\mathrm{d}y\,\mathrm{d}D$$

where MSE = Mean Squared Error

- This quantity tells us not only how good a model is when we have a given dataset D but for **any** dataset with the same number of samples. This is a **very important** notion to keep in mind.
- We **don't want our model to change** its behavior abruptly when we change the samples in dataset, i.e. it should be robust to different choice of training data points. This is not what we would get in that polynomial model of order 100 from before.

-- **The bias-variance decomposition**

It is possible to show that the MSE can be decomposed into three parts:

$$\mathrm{MSE}=\mathbb{E}_{x,D}\left[\left(\hat{f}_D(x)-\mathbb{E}_D[\hat{f}_D(x)]\right)^2\right]+\mathbb{E}_x\left[\left(\mathbb{E}_D[\hat{f}_D(x)]-f(x)\right)^2\right]+\mathbb{E}_{x,y}\left[\left(f(x)-y\right)^2\right]$$

where

1. The first term describes the **variance** of the model: how does the estimate with a given dataset D varies around its mean across all datasets for all values of x.
    - When $\hat{f}_D$ becomes more and more complex (e.g. the number of coefficients in the polynomial model increases) its variance increases, since it becomes capable of adapting to every change in the dataset D and, therefore, has more variability
2. The second term describes the **bias** of the model: how far is the average estimator along different choices of datasets to the actual expected value of the model (i.e. without noise)
    - When $\hat{f}_D$ becomes more and more complex, it is also capable of approximating more closely all of the samples in the dataset, so the bias decreases.
3. The third term describes the **intrinsic noise** of the model and is directly related to the variance of the error term ε

A link to the demonstration of this decomposition is available in the course website.

https://www.cs.cornell.edu/courses/cs4780/2018fa/lectures/lecturenote12.html

My main goal with presenting this decomposition is to show you that the quality of a model can be decomposed into **two main parts**: its variance and its squared-bias.

> See **Figure 4** with the MSE according to model complexity

-- **Training versus testing error**

Our next question is: **how to estimate** the model error that we have juste defined ?

- First, remember that for $x_i \sim p(x)$ sampled IID, we by the law of large numbers that

$$\frac{1}{n}\sum_{i=1}^{n} f(x_i) \to \mathbb{E}_{x\sim p(x)}\big[f(x)\big]$$

With that said, a natural first guess would be to define an **empirical estimator** of the error using the data points $(x_i, y_i) \in D$ that we have access to:

$$\text{eMSE}(\hat{f}_D; D) = \frac{1}{n}\sum_{i=1}^{n}\Big(y_i - \hat{f}_D(x_i)\Big)^2$$

with the hopes of having some sort of convergence to MSE as $n \to \infty$.

- the D after the ";" indicates that the empirical estimator is calculated using data points from D

However, there are **two main problems** with this estimator.

1. The first one is that this quantity concerns only **one dataset** D, so we can't expect it to converge to an expectation over all possible datasets.

   We could consider a setting where we have several datasets $D_1, D_2, \dots, D_N$ and obtain an estimator for each one of them and then average their empirical estimators, we have

$$\frac{1}{N}\sum_{j=1}^{N}\Big(\text{eMSE}(\hat{f}_{D_j}; D_j)\Big) \to \mathbb{E}_{D\sim p^n}\Big[\text{eMSE}(\hat{f}_D; D)\Big]$$

2. The second problem is that even if we average over different datasets, we **will not** have

$$\text{eMSE}(\hat{f}_D; D) = \frac{1}{n}\sum_{i=1}^{n}\Big(y_i - \hat{f}_D(x_i)\Big)^2 \nrightarrow \mathbb{E}_{(x,y)\sim p(x,y)}\Big[\big(y - \hat{f}_D(x)\big)^2\Big]$$

   even though it could look like a case for the law of great numbers to apply.

   What is going on? The problem here is that the there is an **intrinsic statistical dependency** between the samples (x_i, y_i) and the model $\hat{f}_D$ since it is estimated from them. As such, the terms in the average of squares are not exactly IID, meaning that even with $n \to \infty$ we don't have any reason to expect that it should converge to the expectation on the right.

   In fact, the empirical estimator define as above is what we call the **training error** and it has the tendency to always decrease with the complexity of the model.

> See **Figure 5** with the training-testing error in general
>
> See **Figure 6** with the training-testing error for the polynomial model

- The training MSE (in black) is constantly decreasing as the flexibility augments,
- Whereas the testing MSE (red) decreases until a certain point and then starts augmenting again
- This is an indication of **overfitting**: the higher-order polynomials are so flexible that they actually try to "explain the noise" in the training dataset.
  - Note that if you use just the training error, you will never be able to detect overfitting !

But how to have an estimator that **converges** to the true test error?

-- **Split-validation**

It might **sound absurd** to aim for the value of testing error since it depends on data points that we have not seen yet. Nevertheless, we can still estimate it using what people call *cross-validation*.

The idea is pretty simple:

1. Partition the dataset $D = D_T \cup D_V$ into a training set and a validation set

$$D_T = \{(x_{i,T}, y_{i,T})\}_{i \leq n_T} \quad \text{and} \quad D_V = \{(x_{i,V}, y_{i,V})\}_{i \leq n_V}$$

2. Estimate the coefficients of your model using the data points from D_T so we have $\hat{f}_{D_T}$
3. Estimate the error of the model on data points from D_V, which are IID and independent from $\hat{f}_{D_V}$

$$\text{eMSE}(\hat{f}_{D_T}; D_V) = \frac{1}{n_V} \sum_{i=1}^{n_V} \left(y_{i,V} - \hat{f}_{D_T}(x_{i,V}) \right)^2 \to \mathbb{E}_{(x,y)\sim p(x,y)} \left[\left(y - \hat{f}_{D_T}(x) \right)^2 \right]$$

   with convergence when $n_V \to \infty$

This is what people call a **split-validation** procedure.

Note that we always have:

- $n = n_V + n_T$ and, therefore,
  - If n_V the estimation of the test error will have large variance (i.e. not very good)
  - If n_T the coefficients obtained for our model won't be well estimated (i.e. poor estimation)
- **Conclusion**: there is a tradeoff between the sizes of training and validation sizes.
  - In general, we fix $n_V = 0.20 \times n$ and $n_T = 0.80 \times n$

-- **Cross-validation**

Note, however, that **if we stop here** we will be approximating just the MSE for an estimator obtained on a fixed dataset $D_T \subset D$.

A very common extension to split-validation is called **cross-validation** and goes as follows:

1. Split the dataset D in K folds as in the figure

$$D = D_1 \ \cup \ \ldots \ \cup \ D_K$$

   > Draw a figure with folds for a dataset

2. For $k = 1, \ldots K$, consider $D_V^{(k)} = D_k$ and $D_T^{(k)} = \bigcup_{i \neq k} D_i$ and obtain an estimator for the testing error.
3. Calculate the average of the estimated test errors accross diferent folds,

$$\frac{1}{K} \sum_{k=1}^{K} \text{eMSE}(\hat{f}_{D_T^{(k)}}; D_V^{(k)}) \to \mathbb{E}_{D \sim p^n(x,y)} \, \mathbb{E}_{(x,y)\sim p(x,y)} \left[\left(y - \hat{f}_D(x) \right)^2 \right]$$

4.

The procedure explained above is the **standard one** used in practical data science.

Although it might look simple and without too much for it to go wrong, it is a **very delicate** method that is very often used with errors.

We will see **examples of this in the TPs** and later in the course.

-- **Model selection**

Now that you know how to get an **estimate of the generalization error** of a model from data, we can talk about model selection in more precise terms.

Suppose that we were given a dataset with p predictors and we want to know whether a **linear model using only a subset** of them should be preferred.

> Figure with example on the `mtcars` dataset

There are mainly **three strategies** for reducing the number of parameters of a model:

- *Subset selection*: we **identity a subset** of the p predictors that we think is the most related to the response and then fit a linear modeal with least squares to it
- *Shrinkage*: we fit a model to all p predictors but using an extended loss function for the least-squares problem of fitting the parameters. This extension is called a **regularizer** and aims to reduce to zero the parameters of the predictors that are not important for describing the response (we talked a bit about it last week on TD)
- *Dimension reduction*: this approach involves **projecting** the p predictors to a lower-dimensional space defined by linear combinations of variables. A very **well known method** to do this kind of thing is the principal component analysis (aka **PCA**) and we will talk about it next week

We will focus on **subset selection** today.

The **simplest way** of doing subset selection is to **consider all 2^p possible models** with subsets of the predictors and then choose the one which has the smallest cross-validated prediction error.

- The problem with this strategy is that for $p > 10$ things start to become expensive (1024 models to evaluate) to calculate and for $p > 30$ simply untractable (more than 1 billion models to evaluate).

A **more appealing** method for subset selection searches for subsets by doing a **forward stepwise selection**, where we incrementally augment the linear model using the coefficients from a previous step

The procedure goes as follows:

1. Let $\mathcal{M}_0$ denote a model with no predictors (i.e. just the intercept)
2. For $k = 0, \dots, p-1$ we do:

   (a) Consider all $p - k$ models that augment the predictors in $\mathcal{M}_k$ with one extra predictor

   (b) Choose the **best** amont these $p - k$ models and call it $\mathcal{M}_{k+1}$
   NB: Best is defined as having the **smallest training** or testing error. How so?
3. Select a single best model among $\mathcal{M}_0, \dots, \mathcal{M}_p$ using cross-validated prediction error
   NB: Why here we really **should not** compare the models in terms of training error?

This **amounts** to $1 + p(p+1)/2$ models **to evaluate** which is much smaller than 2^p.

- With $p = 20$ we have only ~ 200 models to check, as compared to $\sim 10^6$ in the other method

NB: There exists other options to assess the model error instead of cross-validation, like **AIC and BIC**. They are usually **faster to calculate**. These are scores decrease as the RSS (training error) decreases but are compensated by the increase in the model complexity (i.e. the number of coefficients). We **won't be using** them in the course

> Figure with example of forward stepwise selection on `mtcars` dataset