

# Regression

John Whitamore

## Abstract

Regression models clearly explained with examples in Python.

## 1 Introduction

This paper develops an understanding of regression - what it is and how it is approached in widely-used modelling frameworks.

Regression is the task of learning a mapping  $h : \mathcal{D} \rightarrow \mathbf{y}$  from independent data  $\mathcal{D}$  to dependent data  $\mathbf{y}$  in a way that optimises some objective function. Optimising an objective function might involve minimising a loss function, maximising a utility function or - for probabilistic models - maximising likelihood or marginal likelihood.

In practice, the independent data  $\mathcal{D}$  might be unstructured, incomplete and not in a form which we want to use directly for modelling. We very often decompose the function  $h$  into a data pipeline function  $g$ , which transforms our data into a form  $X$  that is amenable for modelling and a regression function  $f$  which maps onto the observed output data  $\mathbf{y}$ . This yields the compositional structure  $g : \mathcal{D} \rightarrow X, f : X \rightarrow \mathbf{y}$ .

Indeed, it is useful to think of a regression pipeline as having three stages. The first stage is responsible for extracting and collecting raw data and transforming it into a shape that is sufficiently clean and structured for the purpose at hand. We can write  $g : \mathcal{D} \rightarrow \mathcal{D}^*$  in which  $\mathcal{D}$  represents raw data,  $\mathcal{D}^*$  represents clean, structured data and  $g$  represents the function (or ETL pipeline) responsible for data extraction, cleaning and structuring.

The second stage transforms structured data into the form that we want to use within our regression model. We can write  $\phi : \mathcal{D}^* \rightarrow X$ . An example might be using scalar  $x$  data to create values for  $x^0, x^1, x^2, \dots$  for use in polynomial regression. In practice, this stage involves multiple functions, each transforming structured data into a distinct feature representation. These functions are called *basis functions*.

Finally, the third stage is the regression itself, in which the model learns the mapping  $f : X \rightarrow \mathbf{y}$ . This is where we will focus our attention.

Overall, we have a compositional pipeline  $g : \mathcal{D} \rightarrow \mathcal{D}^*, \phi : \mathcal{D}^* \rightarrow X, f : X \rightarrow \mathbf{y}$  that goes from raw to structured data, from structured data to transformed data and - only then - from transformed data to observed outputs. It is clear that the data pipeline mapping(s)  $g$  and the transformation function(s)  $\phi$  are at least as important as the regression mapping  $f$ . It is, perhaps, also clear that  $g$  and  $\phi$  could, in principle, be learned (by optimising an objective function) rather than fixed. This last point is somewhat in the spirit of multi-layer neural network architectures which include both  $\phi$  and  $f$  in their processes. A useful task in a world of AI agents would be also to learn and optimise the data pipeline  $g$  rather than treating it as fixed.

## 2 Linear regression

A linear regression model is one in which the function  $f$  is linear, so that  $f(X) = Xw$ , where  $X \in \mathbb{R}^{N \times M}$  is called the *design matrix* and  $w \in \mathbb{R}^M$  is a vector of regression weights (or coefficients). Note that this means that a linear regression model is linear in its parameters,  $w$ . It does not mean that a linear regression model can only be used to fit straight lines to data.

We will also use  $\epsilon \in \mathbb{R}^N$  to denote a vector of residuals  $\epsilon \equiv y - Xw$  and  $\theta$  to denote the parameters of a model.

Before we put our heads down into the details, it is worth looking up to see the bigger picture. We are describing a world in which data pipelines are designed by humans, basis functions and functional forms are carefully selected by a human and the mapping from design matrix  $X$  to observed data  $y$  is linear. It is very far from a world in which machines are learning or displaying artificial intelligence.

### 2.1 Design matrix and basis functions

Each column of the design matrix  $X$  encodes a hypothesis about the data generating process. The values in each column can be raw data values or - more typically - they can be values obtained by applying basis functions  $\phi$  to raw data values. Columns are often referred to as *features*.

The requirement is (from an interpretative point of view) that the hypotheses are distinct from each other and (from a linear algebra point of view) that the columns of the design matrix are linearly independent. The requirement for linear independence means that the columns of the design matrix form a basis for their span. The term *basis function* reflects the idea that each function contributes to a spanning set over the space of possible input transformations.

Basis functions map a column of (clean, structured) data  $d^*$  onto a column of design matrix values  $x$  so that we have  $\phi : d^* \rightarrow x$ . Commonly-used basis functions include 1)  $\phi(d^*) = 1$  (for bias), 2)  $\phi(d^*) = x^k$ , 3) trigonometric functions, 4) radial basis functions, 5) one-hot encoding and so on. Although the design matrix enables a linear mapping from features to outputs, it can encode a very wide range of features and behaviours.

The principal drawbacks of the design matrix mechanism are 1) that the mapping from features to outputs is always linear in its parameters, 2) that the mapping is fixed at the time of designing the model and 3) that we often do not know how many hypotheses we should encode in the design matrix. This last point gives rise to the question of sparsity, which we will discuss in detail.

### 2.2 Probabilistic regression

We have a vector of observed data  $y$  of length  $N$  and a design matrix of features with  $N$  rows. We therefore have  $N$  points of observed data  $\{y_1, \dots, y_N\}$  and  $N$  rows of the design matrix  $\{x_1^T, \dots, x_N^T\}$  which we can match up with each other to give combined feature and observed data  $\{x_n^T, y_n\}$  for row  $n$ .

For some probability density or probability mass function  $p$ , we have for each row  $p(y_n) = p(y_n | x_n^T w, \theta)$  where  $\theta$  represents the parameters of the probability function. Our choice of a suitable probability density or probability mass function will depend on the data which we are modelling. Common choices include the Poisson distribution for modelling non-negative integers, the Gaussian distribution for modelling real values, the Gamma distribution for modelling strictly positive real values and so on.

It is almost always the case in regression modelling that identically the same probability density or mass function is used for all values of  $n$ . It is also commonly assumed that the probabilities for each

value of  $n$  are independent of each other. These two assumptions taken together are referred to as *identically and independently distributed* or, for short, *i.i.d.*.

Under the i.i.d. assumptions, we can write the joint probability of observing all the  $y_n$  values, given the feature values  $\mathbf{x}_n^T$  and the parameter values  $\boldsymbol{\theta}$ , as

$$p(\mathbf{y}|\mathbf{X}, \mathbf{w}, \boldsymbol{\theta}) = \prod_{n=1}^N p(y_n|\mathbf{x}_n^T \mathbf{w}, \boldsymbol{\theta}) \quad (1)$$

## 2.3 Maximum likelihood

What we'd like to do is to find regression coefficients  $\hat{\mathbf{w}}$  and associated parameter values  $\hat{\boldsymbol{\theta}}$  that maximise the joint probability of the observed data. The subtlety is that, in the probabilistic formulation, these coefficients are treated as fixed. To optimise them, we require a function that shares the same numerical values as the joint probability for given inputs, but treats the parameters as variables.

This leads us to define the *likelihood function*  $\mathcal{L}$ , which takes  $\mathbf{w}$  and  $\boldsymbol{\theta}$  as arguments. We write it as  $\mathcal{L}(\mathbf{y}; \mathbf{w}, \boldsymbol{\theta})$ , and typically seek the values of  $\mathbf{w}$  and  $\boldsymbol{\theta}$  that maximise it.

Note that  $\mathcal{L}(\mathbf{y}; \mathbf{w}, \boldsymbol{\theta}) \equiv p(\mathbf{y}|\mathbf{X}, \mathbf{w}, \boldsymbol{\theta})$ . Both expressions yield the same numerical value for the same inputs. The distinction lies in interpretation. In the likelihood formulation, the data is fixed and the parameters vary; in the joint probability, the parameters are fixed and the data varies. This reversal is what justifies maximising the likelihood rather than the joint probability.

## 2.4 Normally distributed residuals

We define *residuals* as the differences between observed values and the values of the fitted model. Selecting a probability distribution is a key choice when building a model. A common choice for real valued observed data is the Gaussian or Normal distribution.

Under the assumption of normally distributed residuals, we can write

$$\begin{aligned} \mathcal{L}(\mathbf{y}; \mathbf{w}, \boldsymbol{\theta}) &= p(\mathbf{y}|\mathbf{X}, \mathbf{w}, \boldsymbol{\theta}) \\ &= \prod_{n=1}^N \mathcal{N}(y_n|\mathbf{x}_n^T \mathbf{w}, \sigma^2) \\ &= \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(y_n - \mathbf{x}_n^T \mathbf{w})^2}{2\sigma^2}\right) \end{aligned}$$

We would like to maximise this expression with respect to the regression weights  $\mathbf{w}$ . In order to do this, it is convenient to take logs of both sides. Note that the natural logarithm function increases monotonically, so the location of the maximum is unaffected by taking logs. This gives us an expression for the log likelihood  $\ell(\mathbf{y}; \mathbf{w}, \boldsymbol{\theta})$ , denoted using a curly lower case  $\ell$ :

$$\begin{aligned} \ell(\mathbf{y}; \mathbf{w}, \boldsymbol{\theta}) &= \sum_{n=1}^N -\frac{1}{2} \log(2\pi\sigma^2) - \frac{1}{2\sigma^2} (y_n - \mathbf{x}_n^T \mathbf{w})^2 \\ &= -\frac{N}{2} \log(2\pi\sigma^2) - \frac{1}{2\sigma^2} \sum_{n=1}^N (y_n - \mathbf{x}_n^T \mathbf{w})^2 \end{aligned}$$

The shape of derivation that we have used is very common across probabilistic models. We have selected a probability mass or density function (as a modelling choice) and used it to write down an expression for the joint probability of observing the data under the model using the i.i.d. assumptions. From there, we have written a likelihood function and then a log likelihood function which we can optimise with respect to the regression coefficients  $w$ .

### Ordinary Least Squares

Inspection of the expression for log likelihood reveals that  $w$  appears in only one place. It means that we just need to find the  $w$  that minimises the sum of squared residuals,  $\sum_{n=1}^N (y_n - x^T w)^2$ . For this reason, a linear regression model with normally distributed residuals is often referred to as the method of *Ordinary Least Squares*.

Defining a vector of residuals  $\epsilon \equiv y - Xw$ , we can write the sum of squared residuals as  $\epsilon^T \epsilon$

$$\epsilon^T \epsilon = (y - Xw)^T (y - Xw)$$

Differentiating with respect to  $w$  and setting the gradient equal to (vector)  $0$  yields the maximum likelihood estimate of the regression weights:

$$\begin{aligned} \frac{d\epsilon^T \epsilon}{dw} &= 2X^T Xw - 2X^T y = 0 \\ \hat{w} &= (X^T X)^{-1} X^T y \end{aligned}$$

We then use these maximum likelihood weights to obtain our regression curve,  $\hat{y} = X\hat{w}$ . The values in our fitted curve are conditional expectations: what value of  $y$  do we expect given some particular value of  $x^T$ ? It means that we can obtain an estimated value of  $y$  given a new (out-of-sample) value of  $x^T$ , which we can use for prediction.

## 2.5 Gram matrix

In order to obtain our maximum likelihood regression weights, we needed to invert the matrix  $X^T X$ . Not all matrices are invertible. This means that we need to understand what  $X^T X$  is (in linear algebra terms) and what it represents, for interpretability.

$X$  has dimensions  $N \times M$ . This means that  $X^T X$  is a square matrix of dimensions  $M \times M$ . A matrix of the form  $X^T X$  is referred to as the inner product of  $X$  with itself. This type of matrix - formed as an inner product of a matrix with itself - is called a *Gram matrix*.

Element  $\{i, j\}$  of  $X^T X$  is the inner product of column  $i$ ,  $x_i$ , with column  $j$ ,  $x_j$ , of  $X$ . Elements of  $X^T X$  represent - in a loose sense - the degree of similarity between the two columns. More precisely, each vector inner product represents the cosine of the angle between the two vectors multiplied by the lengths of each vector. The cosine of the angle is a measure of similarity because a cosine value of 1 means that the vectors point in the same direction whilst a cosine value of 0 means that the vectors are orthogonal (at right-angles) to each other.

Remembering that each column of  $X$  encodes a hypothesis about the observed  $y$  data, the elements of  $X^T X$  provide an indication of how similar pairs of hypotheses are to each other. In linear algebra terms, if the columns are not linearly independent then  $X$  will not have rank of  $M$  (the number of columns of  $X$ ). In other words, it will not be full rank. This will result in not being able to invert  $X^T X$  and OLS regression will fail. In terms of interpretation, a design matrix  $X$  which is not full rank is one that encodes hypotheses which are not all distinct from each other.

This problem is called *multicollinearity*, meaning that some of the hypotheses are too strongly aligned with each other. In practice, hypotheses need to be quite strongly aligned with each other to cause numerical problems with matrix inversion. However, this means that it is easy to construct an OLS regression model that works (in the numerical sense) but whose results (in an interpretative sense) can be questionable. In particular, poorly conditioned  $X^T X$  matrices tend to lead to unstable estimates of  $w$ .

## 2.6 Non-parametric models based on OLS regression

Our discussion of multicollinearity suggests that it would be useful to ensure that our design matrix  $X$  has columns which are orthogonal to each other. Let's follow through the consequences of that by constructing a design matrix with mutually orthogonal columns.

Consider a design matrix  $X$  with dimensions  $N \times M$  whose columns are mutually orthogonal and have been normalized to unit length. Such a matrix is said to have orthonormal columns. If  $X$  is also square, then it is called an orthogonal matrix.

If  $X$  has orthonormal columns then  $X^T X = I_M$ , the  $M \times M$  identity matrix. If  $X$  is also square we also have  $X X^T = I_N$ , the  $N \times N$  identity matrix. Let's use these properties in the context of our OLS regression results. We have

$$\begin{aligned}\hat{y} &= X\hat{w} \\ &= X(X^T X)^{-1} X^T y \\ &= X I^{-1} X^T y \\ &= X X^T y\end{aligned}$$

If  $X$  is square then we have  $X X^T = I_N$  and we obtain the beautiful but useless result  $\hat{y} = y$ . This is not a model but a tautology: "If you tell me the true value of  $y$ , I'll tell you my estimate of  $y$ ". This demonstrates that  $X$  must not be square.

On the other hand, if  $X$  is "tall", with the number of rows  $N$  being larger than the number of columns  $M$ , we have a result which is useful -  $\hat{y} = X X^T y$  - but in which the matrix  $X$  does not appear directly but only as the outer product  $X X^T$ . This suggests that we could define a new matrix  $K = X X^T$ , subject to  $K$  having the required properties, and re-cast our OLS regression simply as  $\hat{y} = K y$ .

In this formulation, the model extracts structure from  $y$  that is implicitly projected into a lower-dimensional subspace with  $M$  features where  $M < N$ , the number of data points, as for our original design matrix  $X$ . The matrix  $K$  is an *orthogonal projection matrix*.

In order to be an orthogonal projection matrix,  $K$  must be symmetric (which implies square), positive semi-definite and idempotent (so that  $K^2 = X X^T X X^T = X I X^T = X X^T = K$  with  $X^T X = I$ ). The condition for idempotency can be stated as a requirement for all eigenvalues to be either 0 or 1 so that the determinant of  $K$  is 0 or 1 so that  $K^a = K \forall a \in \mathbb{N}$ .

Note that if all eigenvalues of  $\Lambda$  are one, then  $K$  becomes the identity matrix and we get the tautological  $y = \hat{y}$  result that we already rejected. That means that at least one of the eigenvalues of  $\Lambda$  must be zero, which in turns means that  $K$  must specifically be singular rather than merely positive semi-definite. It means that observed data  $y$  can be used to fit a curve  $\hat{y}$  but that we can never fully recover the observed data from the fitted curve. It shows that the fitted curve represents a form of compression.

A mechanism to create suitable  $K$  matrices is to write  $K = Q\Lambda Q^T$  such that a)  $Q$  is  $N \times N$  and orthogonal, with  $Q^T Q = I_N$  and b)  $\Lambda$  is  $N \times N$  and diagonal with diagonal entries in  $\{0, 1\}$ . The number of 1s in  $\Lambda$  determines the rank of  $K$ , i.e. the dimension of the subspace, which must always be less than  $N$ . These conditions are sufficient to guarantee that  $K$  is symmetric, singular and idempotent.

It is possible to write well-known non-parametric models in the form that we have found, with  $\hat{y} = K\mathbf{y}$  and  $K = Q\Lambda Q^T$ . For example, a  $4 \times 4$  Haar wavelet has  $Q$  matrix as below, in which the top row represents scale and subsequent rows represent finer detail.

$$Q = \frac{1}{2} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & -1 \end{bmatrix}$$

### 2.6.1 Ridge Regression

- not positive definite -