Lecture 1: Regression

Iain Styles

7 October 2019

Introduction

Regression is the task of finding the relationship between one or more numerical inputs (*independent* variables) and one or more numerical outputs (*dependent* variables). The goal is to find a relationship (i.e. a mathematical function) that allows us to (predict) the dependent variables from the independent variables, and specifically to learn that relationship from data. As shown in Figure 1, this can, in its simplest form, be described as "curve fitting", but it is a powerful and instructive problem to study because it is very amenable to mathematical analysis which will allow us to understand a range of core issues that are important across the spectrum of machine learning tasks, for example:

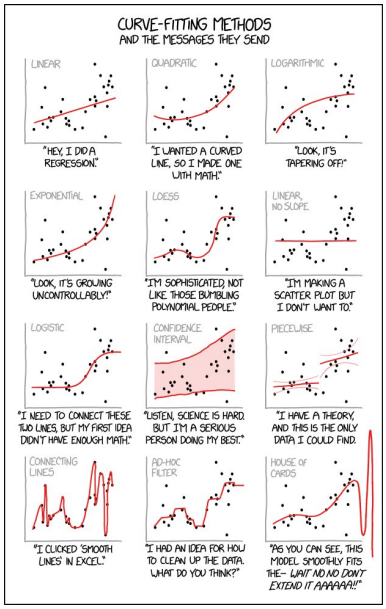

Figure 1: Some approaches to curve fitting. `https://t.co/X76aDTJXI9`

- Objective functions;
- Over/underfitting;
- Regularisation;
- Model capacity;
- Bias vs variance;
- Cross-validation;
- Probabilistic reasoning.

The desired outcomes of a regression are typically to (a) be able to predict values of the dependent variables given values of the independent variables; (b) derive estimates of the parameters that define an underlying mathematical model. This second point is sometimes overlooked but can be an important one – machine learning is often framed as a problem only of prediction but can be used to estimate parameters. Some simple and common examples include the estimation of chemical reaction rates, radioactive half-lives, or vehicle speeds from measurements of their position.

Consider the simple dataset shown in Figure 2 in which we have an independent variable x upon which y is dependent. In this simple example there are two questions we can ask:

1. What is the value of y at $x = 2.5$ (or 3.2, or 1.9 etc)?
2. Given that it seems reasonably obvious that y has a linear dependence on x, what are the parameters (intercept, gradient) of the underlying linear model?

How to answer these questions will be the focus of this part of the course.

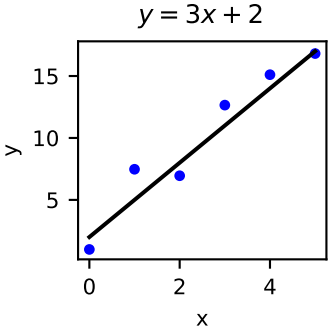

Figure 2: A simple example of a dataset with independent variable x and dependent variable y.

Linear regression

In the first instance, let us consider a class of problems that are *linear*. We will define what we mean by this shortly. In the first instance, we will restrict ourselves to the simple case of data that has one independent variable x and one dependent variable y. A dataset is then comprised of a set of N pairs of data points

$$\mathcal{D} = \{(x_1, y_1), \ldots, (x_n, y_n)\} = \{(x_i, y_i)\}_{i=1}^{N} \tag{1}$$

We wish to model the relationship between x and y as a mathematical function $f(\mathbf{w}, x)$ such that $y_i \approx f(\mathbf{w}, x_i)$ with unknown parameters $\mathbf{w}$. We will ideally choose the form of f so that it accurately describes the underlying data generating process so that all observations of y can be accurately predicted by f. However, this is complicated by the fact that observations and measurements are nearly always subject to some form of error or noise such that there will be a random component of the measurement. That is, if f fully describes the underlying process by which the data is generated, the observations are generated by a noisy process that can be modelled by

$$y_i = f(\mathbf{w}, x_i) + \epsilon \tag{2}$$

where ϵ is a random number drawn from some continuous probability density function that depends on the particular properties of the observation process. We will revisit the implications of this when we consider regression from a probabilistic perspective.

We will first consider a simple way to approach regression by treating it as an optimisation problem in which the objective is to find the value of $\mathbf{w}$ (denoted $\mathbf{w}^*$) that minimises some "loss", or objective function $\mathcal{L}(\mathbf{w})$.

$$\mathbf{w}^* = \arg\min_{\mathbf{w}} \mathcal{L}(\mathbf{w}) \tag{3}$$

The intuition behind this is that $\mathcal{L}(\mathbf{w})$ should be designed to capture the difference between the data and the predictions of the model, and the optimisation will seek to minimise that difference. One very common choice for $\mathcal{L}(\mathbf{w})$ is the *least-squares error*. Given our dataset $\mathcal{D}$ and a modelling function $f(\mathbf{w}, x)$, we construct, for each datapoint in $\mathcal{D}$ a *residual error* defined as

$$r_i(\mathbf{w}) = y_i - f(\mathbf{w}, x_i). \tag{4}$$

This is illustrated in Figure 3.

The least squares error (LSE) loss function is then defined in terms of the residuals as

$$\mathcal{L}_{\text{LSE}}(\mathbf{w}) = \sum_{i=1}^{n} r_i^2 = \mathbf{r}^{\mathrm{T}}\mathbf{r} \tag{5}$$

It is important to note here that $\mathcal{L}_{\text{LSE}}$ has no upper bound, but it does have a finite lower bound because it is a strictly positive

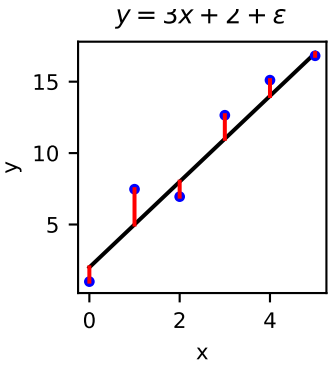

Figure 3: The residuals (shown in red) are a measure of the goodness-of-fit of a function (black line) to a set of data (blue points).

quantity. It is therefore possible to minimise this quantity and, following Equation (3), our goal is to find $\mathbf{w}$ that minimises the loss:

$$\mathbf{w}^* = \arg\min_{\mathbf{w}} \mathcal{L}_{\text{LSE}}(\mathbf{w}) \tag{6}$$

Optimisation problems can be extremely difficult and we will, for now, restrict ourselves to a specific case which can be analysed using some relatively straightforward mathematics: models which are *linear*. What we mean by this is not the familar "$y = mx + c$" example (althought this *is* a linear problem) that you may have seen before, but a more general class of models that are *linear in their unknown parameters*. These will turn out to be surprisingly powerful. Linear models take the form

$$f(\mathbf{w}, x) = w_0\phi_0(x) + \cdots + w_{M-1}\phi_{M-1}(x) = \sum_{i=0}^{M-1} w_i\phi_i(x). \tag{7}$$

Our function is a *linear combination* of a set of *basis functions* $\{\phi_i(x)\}_{i=0}^{M-1}$ weighted by the free parameters $\{w_i\}_{i=0}^{M-1}$. Note that the basis functions have no free parameters and depend solely on the independent variable. The basis functions are a representation of the problem and should be chosen carefully to be sufficiently expressive to capture the features in the data. A very common choice of basis is the polynomials $\left\{x^i\right\}_{i=0}^{M-1}$ which we will use in the examples that follow.

For a finite set of data points $\mathcal{D}$, we can rewrite Equation (7) in matrix form by defining matrix $\mathbf{\Phi}$ with components $\Phi_{ij} = \phi_j(x_i)$ in terms of which

$$\mathbf{f}(\mathbf{w}) = \mathbf{\Phi}\mathbf{w} \tag{8}$$

where the dependency on x is now absorbed into the components of $\mathbf{f}$. It is important to note the order of the indices in the definition of Φ_{ij}: each row (indexed by i) corresponds to a single data point, whilst each column corresponds to a basis function. As an example for a simple quadratic model $f(\mathbf{w}, x) = w_0 + w_1x + w_2x^2$ with basis functions $\left\{x^0, x^1, x^2\right\} = \left\{1, x, x^2\right\}$, we have

$$\mathbf{\Phi} = \begin{pmatrix} 1 & x_1 & x_1^2 \\ 1 & x_2 & x_2^2 \\ & \vdots & \\ 1 & x_N & x_N^2 \end{pmatrix} \tag{9}$$

Having restricted ourselves to linear models, we can begin to solve the optimisation problem posed by Equation (3). The residuals defined by Equation (4) can be written as

$$\mathbf{r} = \mathbf{y} - \mathbf{\Phi}\mathbf{w}, \tag{10}$$

and the loss function (Equation (6)) becomes

$$\mathcal{L}_{\text{LSE}}(\mathbf{w}) = (\mathbf{y} - \mathbf{\Phi}\mathbf{w})^{\text{T}} (\mathbf{y} - \mathbf{\Phi}\mathbf{w}) \tag{11}$$

We now take advantage of our observation that $\mathcal{L}_{\text{LSE}}$ has no upper bound but does have a lower bound to solve Equation (3). To minimise $\mathcal{L}_{\text{LSE}}$ we find the point at which its gradient with respect to its free parameters is zero. Since $\mathcal{L}_{\text{LSE}}$ has no maximum, this point must be the minimum. We therefore differentiate $\mathcal{L}_{\text{LSE}}$ with respect to $\mathbf{w}$ and set to zero. We get

$$\frac{\partial \mathcal{L}_{\text{LSE}}(\mathbf{w})}{\partial \mathbf{w}} = -2\mathbf{\Phi}^{\text{T}} (\mathbf{y} - \mathbf{\Phi}\mathbf{w}) . \tag{12}$$

To understand how we obtain this result let us break the calculation down step-by-step. Noting that $\mathcal{L}_{\text{LSE}}(bw) = \mathbf{r}^{\text{T}}\mathbf{r}$, we first compute the derivative of $\mathbf{r}$ with respect to the components of $\mathbf{w}$. We first note that

$$r_i = y_i - \sum_j \Phi_{ij} w_j \tag{13}$$

Then, picking a particular component of $\mathbf{w}$, say, w_k, to differentiate with respect to, we find that

$$\frac{\partial r_i}{\partial w_k} = -\Phi_{ik}. \tag{14}$$

Now, we note $\mathcal{L}_{\text{LSE}} = \sum_i r_i^2$ and so

$$\frac{\mathcal{L}_{\text{LSE}}}{\partial r_l} = 2r_l \tag{15}$$

Then, we apply the chain rule of differentiation:

$$\frac{\partial \mathcal{L}_{\text{LSE}}}{\partial w_k} = \sum_l \frac{\mathcal{L}_{\text{LSE}}}{\partial r_l} \times \frac{\partial r_l}{\partial w_k} \tag{16}$$

$$= -\sum_l 2r_l \Phi_{lk} \tag{17}$$

Since $\mathbf{r}$ is a *column* vector, we have to do some rearrangements to rewrite this in matrix notation. This means we have to make sure that i) $\mathbf{r}$ is the last term in the equation, and ii) that the matrix $\mathbf{\Phi}$ is in the correct orientation for the multiplication. Writing the result as a vector $\frac{\partial \mathcal{L}_{\text{LSE}}}{\partial \mathbf{w}}$ with components $\frac{\partial \mathcal{L}_{\text{LSE}}}{\partial w_k}$, we have:

$$\frac{\partial \mathcal{L}_{\text{LSE}}}{\partial w_k} = \sum_l -2r_l \Phi_{lk} = -2\sum_l \Phi_{kl}^{\text{T}} r_l \quad \rightarrow \quad \frac{\partial \mathcal{L}_{\text{LSE}}}{\partial \mathbf{w}} = -2\mathbf{\Phi}^{\text{T}}\mathbf{r} = -2\mathbf{\Phi}^{\text{T}} (\mathbf{y} - \mathbf{\Phi}\mathbf{w}) . \tag{18}$$

Finally, we set the result to zero to obtain

$$\mathbf{\Phi}^{\text{T}}\mathbf{y} - \mathbf{\Phi}^{\text{T}}\mathbf{\Phi}\mathbf{w}^* = 0 \tag{19}$$

This result is known as the **normal equations** and is a set of simultaneous linear equations that we can solve for $\mathbf{w}^*$. A naïve way to do this is to evaluate $\mathbf{w}^* = (\mathbf{\Phi}^{\text{T}}\mathbf{\Phi})^{-1}\mathbf{\Phi}^{\text{T}}\mathbf{y}$, but numerical inversion of matrices can be troublesome, especially if the matrix is

large (in this case, large M) and this is best avoided. It is therefore usual to solve the normal equations directly (eg using Gaussian elimination).

This set of mathematical procedures comrise a method by which the parameters of some model can be *learned from data*. This is the very core of what machine learning is about. Although we have set the general form of the model, it is from the data that we learn what its precise form is.

Let us work through a simple example. This can be found in the accompanying notebook which can be accessed at `https://colab.research.google.com/drive/1sHZqzkiDpLgJJmCOodGFo6D4NF9fCgIu`

Reading

Sections 1.1 and 3.1 of Bishop, Pattern Recognition and Machine Learning.