Polar Bear Location Regression Modelling Using Isotopic Data

Kevin Koehler

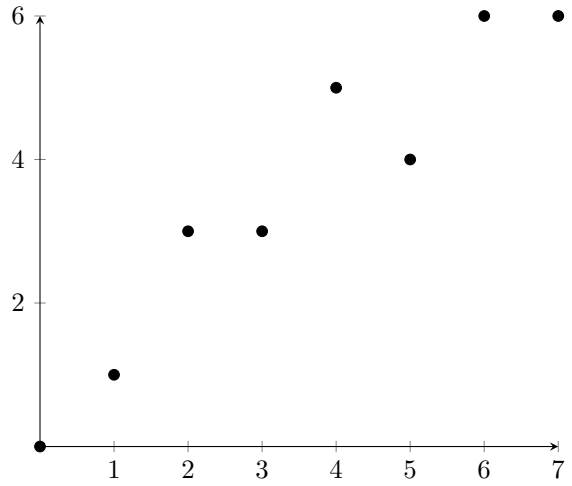
1 Linear Regression Models

1.1 Background

1.1.1 Simple Linear Regression

Regression is about function approximation. Many times, we have some event data described by independent variables called *features* and some dependent variables called *results*. In regression, and indeed machine learning, we assume that there is some function f : features $\rightarrow$ results . In general, we say that where $\mathbf{x}$ is the feature vector and $\mathbf{y}$ is the result vector, we wish to approximate $f(\mathbf{x}) = \mathbf{y}$ with another function that we learn. This function f' takes the form $f'(\mathbf{x}) = \hat{\mathbf{y}}$, where $\hat{\mathbf{y}}$ is our predicted value, which we attempt to get as close to the true value $\mathbf{y}$ as possible.

Linear regression is a regression model that assumes f is approximately linear on its inputs. The word *linear* here actually means hyperplanar, where, if our feature space is n-dimensional, our aim is to produce a n-dimensional *hyperplane* which approximates f. For example, suppose we have a one dimensional feature space with the following data:



Our *line of best fit* can be described by the equation $\hat{y} = mx + b$ where m is the slope and b is the y-intercept. The *least squares* approach finds the line of best fit by minimizing the total error, where the error is the difference between

the true value and the predicted value. Let $\hat{\epsilon_i}$ be the difference between $\hat{y}$ and y_i, i.e. $\hat{\epsilon_i} = |mx_i + b - y_i|$. Our goal, then, is to find minimal m, b for all $\sum_i \epsilon_i$.

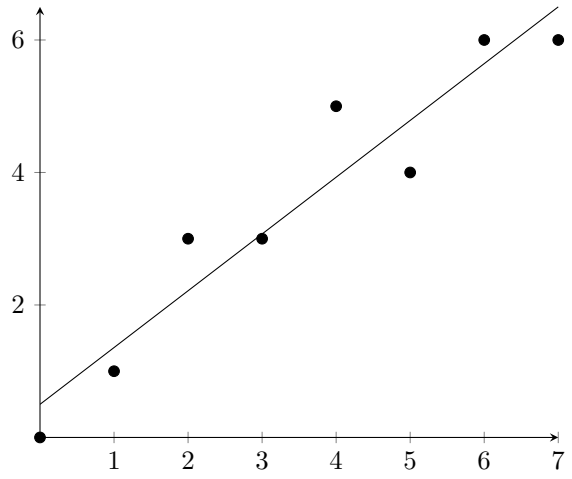
We can find minimal values m, b with the following recursive algorithm:

$$b = \bar{y} - m\bar{x}$$
$$m = \frac{\sum_i (x_i - \bar{x})(y_i - \bar{y})}{\sum_i (x_i \bar{x}^2}$$

With an R^2 value of:

$$r_{xy} = \frac{\bar{xy} - \bar{x}\bar{y}}{\sqrt{(\bar{x^2} - \bar{x}^2)(\bar{y^2} - \bar{y}^2)}}$$

If we do the calculations as above for our simple example, we arrive at values $m = 0.857143, b = 0.5$, with R^2 value of 0.907563. This results in a graph that looks like this:



1.1.2 Multiple Linear Regression

Very often, we have k independent variables rather than one. This means that instead of constructing a line to fit the data, we must construct a k-dimensional plane. For reason that will become clear, in machine learning we often denote the equation of the plane as:

$$\hat{y} = w_0 + w_1 x_1 + ... + w_k x_k$$

In regression modelling, we are trying to fit our data. From the data, for regression problems, which are *supervised* learning problems, the data must contain both the independent variables $\mathbf{X}$ and the results $\mathbf{y}$. Thus the our model becomes:

$$\hat{\mathbf{y}} = \mathbf{wX}$$

Here $\mathbf{w}$ is the weight vector which we attempt to optimize such that the error is minimized. The error can be calculated using the *mean squared error*:

$$MSE(\mathbf{y}, \hat{\mathbf{y}}) = \frac{1}{k}\sum_{i=1}^{i=k}(y_i - \hat{y}_i)$$

Let $\mathbf{A}_i$ denote the ith row of a matrix $\mathbf{A}$. Thus the value we must optimize is:

$$E = \sum_{j=1}^{j=n} MSE(\mathbf{X}_j\mathbf{w}, \mathbf{y}_j)$$

We can use the *ordinary least squares* (OLS) method to find the weight vector. To find the weight vector $\mathbf{w}$ we are looking for a vector $\hat{\mathbf{w}}$ which satisfies:

$$\hat{\mathbf{w}} = \underset{\hat{\mathbf{w}}}{\operatorname{argmin}} \sum_{i=1}^{i=k}(y_i - w_0 - (\hat{\mathbf{w}}\mathbf{X})_i)^2$$

We can find the solution to the above problem with the following equation:

$$\hat{\mathbf{w}} = (\mathbf{X}^\intercal\mathbf{X})^{-1}\mathbf{y}\mathbf{X}^\intercal$$

1.1.3 Multivariate Linear Regression

Multivariate linear regression deals with the case where the result vector $\hat{\mathbf{y}}$ has entries with length $m > 1$, i.e. it is a matrix.

Let $\mathbf{X}$ be the matrix of features, consisting of n observational records of length k, thus taking the shape $n \times k$. Let $\mathbf{Y}$ be the matrix of results, consisting of n observational records of length m, thus taking the shape $n \times m$. Then our more complex model will take the form:

$$\hat{\mathbf{Y}} = \mathbf{X}\mathbf{W}$$

Where $\hat{\mathbf{Y}}$ is the $n \times m$ matrix of predicted values, $\mathbf{W}$ the $n \times m$ weight matrix we wish to optimize. In the previous subsection, we optimized a single weight vector. In multivariate regression, we aim to optimize a single weight vector $\mathbf{w}$. Here we aim to optimize the row vectors $\mathbf{W}_i$ of $\mathbf{W}$ such that the following total error of the model is minimized:

$$\begin{aligned} E &= \sum_{i=1}^{i=n} MSE((\mathbf{X}\mathbf{W})_i, \mathbf{Y}_i) \\ &= \sum_{i=1}^{i=n} MSE(\hat{\mathbf{Y}}_i, \mathbf{Y}_i) \end{aligned}$$

Again, we use the OLS method to find a weight matrix $\hat{\mathbf{W}}$ such that:

$$\hat{\mathbf{W}} = \underset{\hat{\mathbf{W}}}{\operatorname{argmin}} \sum_{i=1}^{i=n}\sum_{j=1}^{j=m}(\mathbf{Y}_{ij} - \mathbf{W}_{0j} - \mathbf{X}\mathbf{W}_{ij})^2$$

The solution to which being:

$$\hat{\mathbf{W}} = (\mathbf{X}^\intercal\mathbf{X})^{-1}\mathbf{Y}\mathbf{X}^\intercal$$

1.2 Implementation

For the polar bear data, there are two approaches one could take. Firstly, we could use multiple linear regression with the isotopic features to regress the zone for which polar bears were tagged. Another approach would be to use multivariate regression to predict the longitude and latitude values. Both approaches will be explored in this document.

To get results, the python module "sk-learn" was used to optimize the ordinary least squares objective function.

Before fitting the model, all features were scaled.

1.2.1 Multiple Linear Regression

In the data, there are seven zones. These zones were given a label between 1 and 7. The multiple linear regressor was fit to these zones. The model outputs a continuous number, often something like "2.121..." for the purposes of calculating MSE and R^2. These continuous outputs were then rounded to the closest integer, which corresponds to one of the zone labels. This allows for the calculation of accuracy.

1.2.2 Multivariate Linear Regression

The multivariate linear regressor attempts to predict the longitude/latitude of the polar bears based on the feature subsets. The MSE it outputs is the (squared) average deviance of the predicted longitude/latitude values from the true longitude/latitude values. For example, if the true coordinates were $(100, 100)$ and the the model predicted $(90, 90)$ then the MSE would be 100.0.

1.3 Results

The python module "sk-learn" allows for arbitrarily worsening R^2 such that an R^2 score of 0 indicates that the model is randomly guessing, or disregarding the inputs, and a negative R^2 indicates that the model performs worse than random.

The results shown below are 10-fold cross validated.

1.3.1 Results Table

Table 1: Linear Regression Results

	Features		
Outcomes	*d2H, d13C, d15N*	*d15N,d13C,d2H,Year*	*Year*
Longitude/Latitude	MSE: -41.5 R^2: -1.50	MSE: -34.9 R^2: -1.12	MSE: -161 R^2: -0.675
Latitude	MSE: -24.5 R^2:-0.0354	MSE: -11.0 R^2:0.315	MSE: -15.5 R^2: 0.306
Longitude	MSE: -58.5 R^2:-2.55	MSE: -58.8 R^2:-2.55	MSE: -306 R^2:-1.65
Zone	MSE: -1.34 R^2: -0.0354	MSE: -0.903 R^2: 0.29	MSE: -1.25 R^2: 0.125
Zone Accuracy	33.8%	62.0%	29.7%

1.3.2 Results Discussion

From the results, we can gather that neither the multiple linear regressor (classifier) or the multivariate linear regressor are sufficiently accurate enough. The classifier did achieve 62% accuracy, but I suspect this is due to sampling bias, where certain zones were sampled much more frequently in particular years. When the year was taken out of the feature space, the models were never able to obtain a positive R^2 value.

All of the objective functions implemented in the software library I used were tested on the data. None improved results.

I suspect that function we are approximating is indeed linear on its inputs. However, the nature of the multiple linear classifier was perhaps not ideal for the job, as it spits out a continuous latent variable which must be rounded in order to categorize the ouput. To remedy this, perhaps a logistic regression model may produce better results.

If the function is non-linear, a non-linear classifier may be necessary (such as a support vector machine). If the function is not smooth, then most likely a feedforward neural network classifier will be necessary.

2 Logistic Regression Models

2.1 Binary Logistic Regression Model

3 Non-Linear Regression Models

4 Bayesian Classifiers

5 Perceptron Classifiers

6 Feedforward Neural Network Models

7 Advanced Models