MATH 365
Linear Regression Lab

1 Introduction

Linear regression is a way to understand the linear relationship between two or more variables. This relationship can be written as a mathematical formula and used to

- describe the linear dependence of one variable on another.
- predict values of one variable from values of another.
- correct for the linear dependence of one variable on another, in order to clarify other features of its variability.

The **correlation coefficient**, on the other hand, measures the strength of a linear relationship between variables. This is different from regression which focuses on the mathematical form of the relationship.

2 Simple Linear Least Squares Regression

In simple linear regression the mathematical problem is as follows:

$$\text{Given a set of } k \text{ points } (x_i, y_i) \quad i = 1, 2, ..., k$$

assume they are related through the equation

$$y_i \approx b_0 + b_1 x_i$$

where b_0 and b_1 are constant (unknown) coefficients. Most data (x_i, y_i) contain noise and it is not possible for every data point to satisfy this linear relationship. Therefore we seek to find b_0 and b_1 so that

$$y_i = b_0 + b_1 x_i + n_i$$

where n_i represents the noise in each data point.

For least squares regression we assume the noise n_i follows a Gaussian or normal distribution. Our goal is to find coefficients b_0 and b_1 and form the best fit, or **regression line**

$$\hat{y}(x) = b_0 + b_1 x. \tag{1}$$

If the noise has zero-mean we find b_0 and b_1 by minimizing S, the sum of squared errors:

$$S = \sum_{i=1}^{k} (y_i - \hat{y}_i)^2.$$

We can use multivariable calculus to find the values of b_0 and b_1 that minimize S but we will not go through the derivations here. If we let $\mu_{\mathbf{x}}$ denote the mean of $\mathbf{x}$ and $\mu_{\mathbf{y}}$ the mean of $\mathbf{y}$, then the coefficients b_0 and b_1 can be written in the convenient form

$$\begin{aligned} b_0 &= \mu_{\mathbf{y}} - b_1\mu_{\mathbf{x}} \\ b_1 &= \frac{\sum_{i=1}^{k}(x_i - \mu_{\mathbf{x}})(y_i - \mu_{\mathbf{y}})}{\sum_{i=1}^{k}(x_i - \mu_{\mathbf{x}})^2}. \end{aligned}$$

The predicted value of y at any point x is given by $\hat{y}(x)$ in (1). At the data points x_i, $\hat{y}(x_i)$ will most likely not equal the observed value y_i. The residual $e_i = y_i - \hat{y}(x_i)$ measures how far the prediction is from the observations. The variance in the residual is

$$\hat{\sigma}^2 = \frac{\sum_{i=1}^{k} e_i^2}{k-2}. \tag{2}$$

The $k-2$ in the denominator is known as the "degrees of freedom", and is computed by subtracting the number of parameters estimated (b_0 and b_1) from the number of observations.

3 Correlation Coefficient

The **coefficient of determination** r^2 is a measure of how well the regression line represents the data. It is defined as:

$$r^2 = 1 - \frac{\sum_{i=1}^{k} e_i^2}{\sum_{i=1}^{k}(y_i - \mu_{\mathbf{y}})^2.} \tag{3}$$

In simple linear regression, the **correlation coefficient** r is simply the square root of the coefficient of determination. One advantage of r is that it is unitless, allowing researchers to make sense of correlation coefficients calculated on different data sets with different units.

As an example consider $r = 0.9$, i.e. $r^2 = 0.81$. This means that 81% of the total variation in $\mathbf{y}$ can be explained by the linear relationship between $\mathbf{x}$ and $\mathbf{y}$. The other 19% of the total variation in $\mathbf{y}$ remains unexplained.

4 Multivariate Linear Least Squares Regression

Simple linear regression is limited because we can only consider two variables of interest. For example, a child's height (y) may depend on both their mother's (x_1) and father's (x_2) height. In simple linear regression we only have the following two equations to work with

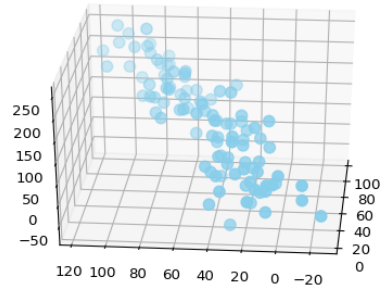
$$\hat{y}(x_1) = b_{10} + b_{11}x_1 \quad \text{and} \quad \hat{y}(x_2) = b_{20} + b_{21}x_2$$

where b_{10}, b_{11} are the constants for simple regression with their mother's height b_{20}, b_{21} and are the constants for the father's. The problem with two separate regression lines is that neither captures how y depends on x_1 **and** x_2 simultaneously. In fact, by not including all *confounding factors* in the simple linear regression equations we create bias in our estimate of the child's height.

Multivariate regression is a step towards remedying this by adding variables to the simple linear regression case, i.e.

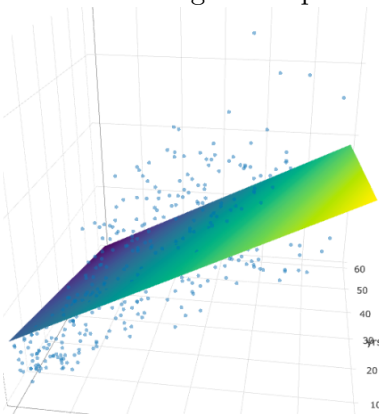
$$\hat{y}(x_1, x_2) = b_0 + b_1x_1 + b_2x_2.$$

A good way to think about regression is that we're finding weights (b_1, b_2) that determine the importance of each of the variables $(x_1,\ x_2)$. To visualize this relationship, we need to think in three dimensions. Before, with simple regression, we could plot the relationship on a 2D graph. Now the data points are in a 3D space of child's height, mother's height, and father's height. Here's a picture of a 3D scatterplot.



Note that this graph does not represent the height data we have been talking about, i.e. $x_1 \in [-20, 120]$, $x_2 \in [0, 100]$ and $y \in [-50, 250]$.

When we run a regression in this space, we find a regression plane, not a regression line. This plane in 3D space has two different slopes: the amount it increases when you increase the x_1 variable and the amount when you increase x_2. The variables b_1 and b_2 describe the two slopes but there is only one intercept b_0 which is the value of the plane where it touches the vertical y-axis. Here is a picture of a different 3D scatterplot with a 2D regression plane.



Visualizing multvariate regression helps our understanding of the process. However, we will not focus on this aspect as it needs more attention than we have time and can be found in a course focused on visualization.

In order to find equations for the multivariate coefficients b_0, b_1 and b_2 we use the matrix form of regression. If we have two independent variables (e.g. mother's and father's height) the data are (x_{i1}, x_{i2}, y_i) $i = 1, \ldots, k$. Each data point satisfies

$$y_i = b_0 + b_1 x_{i1} + b_2 x_{i2} + n_i.$$

All the data can be written simultaneously with the the matrix equation

$$\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_k \end{bmatrix} = \begin{bmatrix} 1 & x_{11} & x_{12} \\ 1 & x_{21} & x_{22} \\ \vdots & \vdots & \vdots \\ 1 & x_{k1} & x_{k2} \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \end{bmatrix} + \begin{bmatrix} n_1 \\ n_2 \\ \vdots \\ n_k \end{bmatrix}.$$

We can write this matrix equation more simply as

$$\mathbf{y} = \mathbf{X}\mathbf{b} + \mathbf{n}. \tag{4}$$

The least squares strategy for finding $\mathbf{b}$ is the same as in the simple linear regression model, but in multi-dimensions. We minimize S, the sum of squared errors

$$S = \sum_{i=1}^{k} (y_i - \hat{y}_i(x_1, x_2))^2 = \sum_{i=1}^{k} (\mathbf{y} - \mathbf{X}\mathbf{b})_i^2.$$

We again use multivariable calculus to find the values in $\mathbf{b}$ that minimize S but we will not go through the derivations here. The coefficients can be written in the convenient form

$$\mathbf{b} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}. \tag{5}$$

The variance in the residual $\hat{\sigma}^2$ and the correlation coefficient r are calculated in the same manner as for simple linear regression in (2) and (3), respectively. The R^2 value in multivariate linear regression is often called the coefficient of multiple determination.

5 Matrices in Python

Multivariate regression requires us to create matrices and solve systems of equations. Vectors such as $\mathbf{v} \in \mathbb{R}^k$ can be thought of as a special case of a matrix like $\mathbf{A} \in \mathbb{R}^{k \times p}$. With that thought in mind creating a matrix in Python is very similar to creating a vector. Recall to create the vector we type

```
v=np.array([1,2,3])
```

This is a row vector $[1, 2, 3]$, i.e. $\mathbf{v} \in \mathbb{R}^{1\times 3}$. Instead, let's create it as a column vector $\begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} \in \mathbb{R}^{3\times 1}$

```
v_col = np.array([[1],
                  [2],
                  [3]])
```

In Python v has shape (3,) while v_col has shape (3, 1).

Viewing the difference between row and column vectors helps us see how a vector is a special case of a matrix. For example the matrix $\mathbf{A} = \begin{bmatrix} 1 & 2 & 3 \\ 3 & 4 & 5 \\ 6 & 7 & 8 \end{bmatrix} \in \mathbb{R}^{3\times 3}$ is created in Python by

```
A = np.array([[1,2,3],
              [3,4,5],
              [6,7,8]])
```

Matrix indexing is similar to vector indexing. For example, $\mathbf{A}_{23} = 5$ and this is accessed in Python with A[1,2]. (Recall that Python indexing starts as 0). If we want to access the last two columns of the first two rows

```
print(A[:2, 1:3])
```

returns

```
[ [2  3]
  [4  5]]
```

Matrix algebra such as addition, subtraction and multiplication requires that the dimension of the matrix and vectors *"agree"*. For addition and subtraction, the objects must have the same dimension. In Python we can simply type v+v or A+A.
WARNING: If we type v+v_col this should give us an error but instead we get

```
[[2,3,4],
 [3,4,5],
 [4,5,6]])
```

I suggest thinking about how Python is interpreting this algebra.

Matrix multiplication such as $\mathbf{A}\mathbf{v}_{col}$ requires that the number of columns in $\mathbf{A}$ equal the number of rows in $\mathbf{v}_{col}$. The process of matrix multiplication involves multiplying each row by each column. If you haven't done this in awhile, I recommend reviewing it. Matrix multiplication in Python is done with the *np.dot()* command, e.g.

```
print(np.dot(A,v_col))
```

produces the correct answer

```
[[14]
 [26]
 [44]]
```

WARNING: If we type print(np.dot(A,v)) this should give us an error (why?) but instead we get

```
[14  26  44]
```

Fortunately, if we type print(np.dot(v_col,A)) we will get an error message while if we type print(np.dot(v,A)) we get the correct answer (what is it?). While Python has advanced capabilities for data analysis, unfortunately it falls short with matrix algebra.

Our focus here is to solve the linear system $\mathbf{X}\mathbf{b} = \mathbf{y}$ for the multivariate regression coefficients. More advanced courses in computational math focus on algorithms to solve linear systems, while in this introductory course we will just the existing library *np.linalg.solve()*.

For example solve $\mathbf{A}\mathbf{v}_{col} = \mathbf{y}$ for $\mathbf{v}_{col}$. First we form $\mathbf{y}$, then use np.linalg.solve to estimate $\mathbf{v}_{col}$:

```
y =np.array( [[14],
              [26],
              [44]])
v_col_est = np.linalg.solve(A,y)
```

print(v_col_est) gives us the estimate

```
[[2.]
 [0.]
 [4.]]
```

Interesting! The correct answer is $\mathbf{v}_{col} = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}$ and the estimate v_col_est is very different. The issue is not that Python is doing something fishy with our matrices and vectors. Rather, the issue is the mathematical solution of $\mathbf{A}\mathbf{v}_{col} = \mathbf{y}$ has issues. We'll talk more about this later.

Let's say we don't know know the correct answer is $\mathbf{v}_{col}$. How can we check if our answer is correct? Naively, we could take our estimate v_col_est, multiply it by A and see if we get y

```
print(np.dot(A,v_col_est))
```

The result is

```
[[14]
 [26]
 [44]]
```

Interesting! The matrix multiplication gave us the correct y even though our estimate v_col_est was incorrect. More on this later.

6 Individual Lab

Simple Linear Regression

These questions should be completed in Blackboard before class. For the data (0,0), (-1,1) and (4,-1) answer the following questions in Blackboard

1. Find the regression coefficients b_0 and b_1.
2. Find the predicted value of y at $x = 2$.
3. Find the residual $e_2 = y_2 - \hat{y}(x_2)$.
4. Find the variance in the residuals.
5. Find the correlation coefficient between between $\mathbf{x}$ and $\mathbf{y}$.

Multivariate Linear Regression

1. Consider the multivariate data (x_1, x_2, y): $(-2, 4, 1), (-2, 1, 0), (0, 0, 2), (1, 1, 3)$ and assume we fit it to the line $\hat{y}(x_1, x_2) = b_0 + b_1 x_1 + b_2 x_2$.

   (a) Identify the matrix $\mathbf{X}$ in (4).

   (b) Identify the vector $\mathbf{y}$ in (4).

   (c) Identify the formula for the coefficients $\mathbf{b} = \begin{bmatrix} b_0 \\ b_1 \\ b_2 \end{bmatrix}$ in (5).

   (d) The estimates of the coefficients are $b_0 = 1.83, b_1 = 1, b_2 = 0.28$. How would you interpret these values?

2. If $\mathbf{X} = \begin{bmatrix} 1 & -2 & 4 \\ 1 & -2 & 1 \\ 1 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$ then the following values of $\mathbf{b}$ and $\mathbf{y}$ satisfy $\mathbf{Xb} = \mathbf{y}$.

7 Group Work

Simple Linear Regression

Please write your answers to the following questions on the supplied note cards.

1. Given the data (0,0), (-1,1) and (4,-1) write the commands that create Numpy arrays x and y containing the x and y coordinates, respectively.

2. Given only a Numpy array x, write a function that computes the mean of the elements in x. Please complete the remaining questions in groups of 2-3 people.

3. Using only pen and paper, create a function regcoef(x,y) that returns the linear regression coefficients b_0 and b_1. Use these tips and be prepared to explain to the rest of the class why one should follow them.

   (a) First write a function to compute the mean and call it within regcoef

   (b) Form b_1 before b_0.

   (c) Form the numerator and denominator in b_1 separately

   (d) Call the function that computes the mean first.

4. Code your functions from 3 in Python and test them by finding the linear regression coefficients for the data (0,0), (-1,1) and (4,-1). Use the answer you found in the homework due today to check that your function is working correctly. Be prepared to discuss with the class the errors you encountered and your debugging strategies.

5. Define two anonymous functions: one to compute $\hat{y}$ and a second to compute residuals e_i. Define the functions so that you can form the residual vector $\mathbf{e}$ with one line. Be prepared to discuss the function inputs and how you would access each element e_i.

6. Write a regular function that computes the variance in the residual and use your anonymous functions from 5. Check that your function works by testing it with data (0,0), (-1,1) and (4,-1) for which you know the correct answer. Be prepared to discuss with the class the errors you encountered and your debugging strategies.

7. Write a regular function that computes the correlation coefficient. Again check your answer with data (0,0), (-1,1) and (4,-1) for which you know the correct answer. In addition, plot the data and the regression line on the same graph. Be prepared to discuss with the class how well the line predicts points other than the data.

Multiple Linear Regression

An important characteristic of a semiconductor product is the pull strength of a wire bond. We will investigate the suitability of using a linear multiple regression model to predict pull strength y as a function of wire length x_1 and die height x_2.

1. Load the data in wire.txt and form one dimensional arrays y, x_1 and x_2. Here are some tips

   (a) Refer to the Working with data files Lab to remind yourself how to read and work with a data file.

   (b) Identify the shape of the array you created when loading the data. All the data will be loaded into one column. Separate it into the correct number of rows and columns by following the instructions on the bottom of page 2 and top of page 3 in the Working with data files Lab.

   Be prepared to discuss the shape of the arrays y, x_1 and x_2 and how they relate to our formula for the regression coefficients $\mathbf{b}$ in equation (5).

2. Form the regression matrix $\mathbf{X}$ in (4). Note that it has 3 columns and here are some tips

   (a) You can transpose a matrix $\mathbf{A}$ by executing A.transpose()

   (b) You can create an array of 1s using np.ones.

   (c) You can stack 1-D arrays as columns using np.column_stack

   Be prepared to discuss challenges in getting the dimension of $\mathbf{X}$ correct.

3. The formula for $\mathbf{b}$ in (5) involves finding the inverse of the matrix $\mathbf{X}^T\mathbf{X}$. Don't use that formula, rather find $\mathbf{b}$ by solving the system $\mathbf{X}^T\mathbf{X}\mathbf{b} = \mathbf{X}^T\mathbf{y}$ because that is a more accurate and efficient approach on a computer.

   (a) Use matrix algebra to explain why forming $\mathbf{b}$ as described in (5) is the same as solving the system $\mathbf{X}^T\mathbf{X}\mathbf{b} = \mathbf{X}^T\mathbf{y}$ for $\mathbf{b}$.

   (b) Form the matrix $\mathbf{A} = \mathbf{X}^T\mathbf{X}$ and the vector $\mathbf{rhs} = \mathbf{X}^T\mathbf{y}$ and use np.linalg.solve to solve $\mathbf{A}\mathbf{b} = \mathbf{rhs}$.

   Be prepared to report the dimension of $\mathbf{A}$ and $\mathbf{rhs}$ and your values for the coefficients $b_i,\ i = 1, \ldots 3$.