Multiple linear regression

Herman Kamper

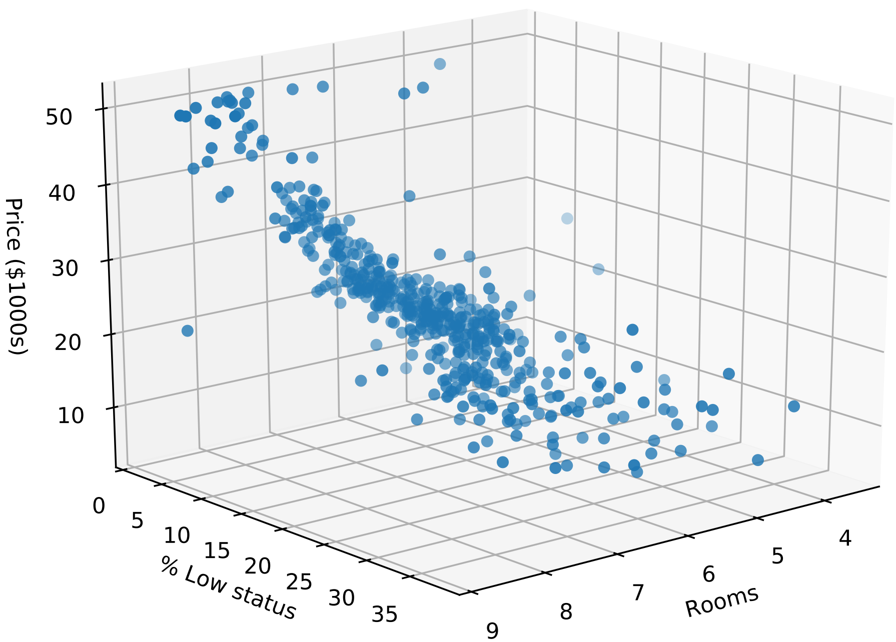
Overview

In simple linear regression we have a single input feature x from which we want to predict a scalar output y.

In multiple linear regression, we instead have several input features, x_1, x_2, ..., x_D, from which we want to predict a scalar output y.

You can think of grouping all the input features into a feature vector $\mathbf{x}$ from which we now want to predict a scalar output y.

Boston house prices



Multiple linear regression

The model

$$f(x_1, x_2, \dots, x_D; w_0, w_1, \dots, w_D) =$$

We can write this in vector form:

$$f(\mathbf{x}; \mathbf{w}) = \mathbf{w}^\top \mathbf{x}$$

where

$$\mathbf{w} = \qquad \text{and} \qquad \mathbf{x} =$$

The loss function

Squared loss:

$$J(\mathbf{w}) = \sum_{n=1}^{N} \left(y^{(n)} - f(\mathbf{x}^{(n)}; \mathbf{w}) \right)^2$$

$$=$$

Optimisation

We want to find the setting of the parameter vector $\mathbf{w}$ that minimises the loss:

$$\hat{\mathbf{w}} = \arg\min_{\mathbf{w}} J(\mathbf{w})$$

Strategy: Set $\frac{\partial J}{\partial w_0} = 0$, $\frac{\partial J}{\partial w_1} = 0$, $\frac{\partial J}{\partial w_2} = 0$, $\ldots$, and $\frac{\partial J}{\partial w_D} = 0$, and solve the equations jointly.

Are you looking forward to deriving these equations one by one and then solving the $D + 1$ equations jointly?

Idea: Rather write everything in vector form and set $\frac{\partial J}{\partial \mathbf{w}} = 0$.

But what does it mean to take the derivative of a function with respect to a vector?

Interlude: Read the note on vector and matrix derivatives.

Writing the loss in matrix form

We want to minimise:

$$\begin{aligned} J(\mathbf{w}) &= \sum_{n=1}^{N} \left(y^{(n)} - f(\mathbf{x}^{(n)}; \mathbf{w}) \right)^2 \\ &= \sum_{n=1}^{N} \left(y^{(n)} - \mathbf{w}^\top \mathbf{x}^{(n)} \right)^2 \end{aligned}$$

Define:

$$\mathbf{X} = \qquad \text{and} \qquad \mathbf{y} \quad =$$

This allows us to write the loss as

$$\boxed{J(\mathbf{w}) = (\mathbf{y} - \mathbf{X}\mathbf{w})^\top (\mathbf{y} - \mathbf{X}\mathbf{w})}$$

This might be somewhat hard to see immediately, so let us just see how we get here. We define an error vector $\mathbf{e}$ as:

$$\mathbf{e} =$$

This allows us to write the loss as

$$J(\mathbf{w}) = \sum_{n=1}^{N} \left(y^{(n)} - \mathbf{w}^\top \mathbf{x}^{(n)} \right)^2 = \mathbf{e}^\top \mathbf{e}$$

Since $\mathbf{e} = \mathbf{y} - \mathbf{X}\mathbf{w}$, we get the loss in matrix form as in the equation above.

So why did we go through all that effort? We now have a form for $J(\mathbf{w})$ containing only matrices and vectors. This mean we can just use our vector and matrix derivatives to determine $\frac{\partial J}{\partial \mathbf{w}}$!

To do this, we just multiply out $J(\mathbf{w})$ a little bit more:

$$J(\mathbf{w}) =$$

$$= \mathbf{y}^\top \mathbf{y} - 2\mathbf{w}^\top \mathbf{X}^\top \mathbf{y} + \mathbf{w}^\top \mathbf{X}^\top \mathbf{X}\mathbf{w}$$

Optimising the loss: The normal equations

Now we set $\frac{\partial J}{\partial \mathbf{w}} = 0$, i.e. $\frac{\partial J}{\partial w_0} = 0$, $\frac{\partial J}{\partial w_1} = 0$, $\ldots$, $\frac{\partial J}{\partial w_D} = 0$.

$$\frac{\partial J}{\partial \mathbf{w}} = \frac{\partial}{\partial \mathbf{w}} \left[\mathbf{y}^\top \mathbf{y} - 2\mathbf{w}^\top \mathbf{X}^\top \mathbf{y} + \mathbf{w}^\top \mathbf{X}^\top \mathbf{X} \mathbf{w} \right]$$

$$=$$

$$=$$

In the second step above we used the following identities from Wikipedia:

$$\frac{\partial \mathbf{x}^\top \mathbf{a}}{\partial \mathbf{x}} = \mathbf{a}$$
$$\frac{\partial \mathbf{x}^\top \mathbf{A} \mathbf{x}}{\partial \mathbf{x}} = (\mathbf{A} + \mathbf{A}^\top)\mathbf{x}$$

Now set $\frac{\partial J}{\partial \mathbf{w}} = 0$:

$$\mathbf{X}^\top \mathbf{X} \mathbf{w} = \mathbf{X}^\top \mathbf{y}$$

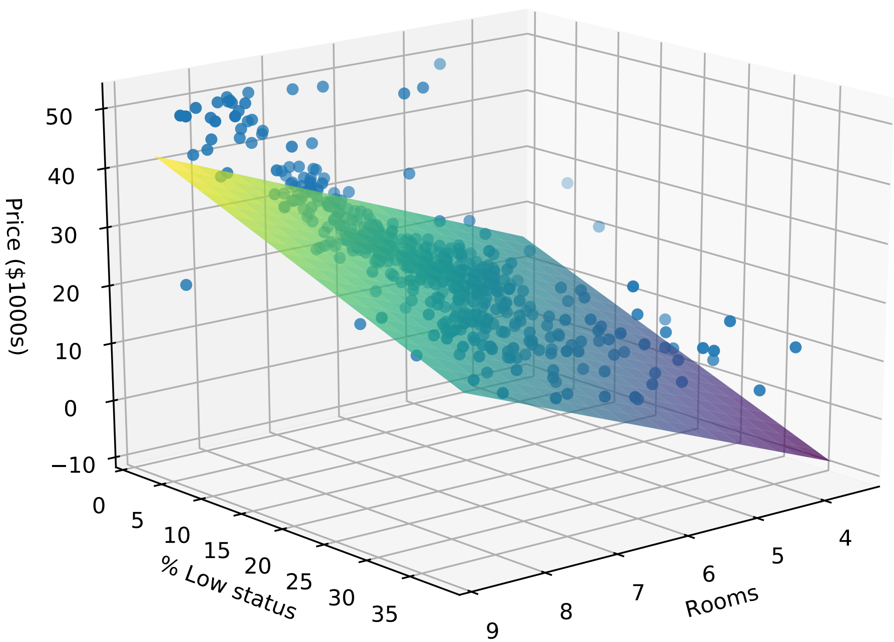
This results in:

$$\boxed{\hat{\mathbf{w}} = \left(\mathbf{X}^\top \mathbf{X}\right)^{-1} \mathbf{X}^\top \mathbf{y}}$$

This is the solution to $D + 1$ equations (the dimensionality of $\mathbf{w}$). This set of equations is called the *normal equations*.

The amazing thing here is that we can get this optimal setting $\hat{\mathbf{w}}$ using just one line of Python code!

Boston house prices fit



$$f(\mathbf{x};\hat{\mathbf{w}}) = \hat{w}_0 + \hat{w}_1 x_1 + \hat{w}_2 x_2 = -1.358 + 5.095x_1 - 0.642x_2$$

Videos covered in this note

- Linear regression 2.1: Multiple linear regression - Model and loss (16 min)
- Linear regression 2.2: Multiple linear regression - Optimisation (8 min)

Reading

- ISLR 3.2 intro
- ISLR 3.2.1