Trinity College Dublin
Coláiste na Tríonóide, Baile Átha Cliath
The University of Dublin

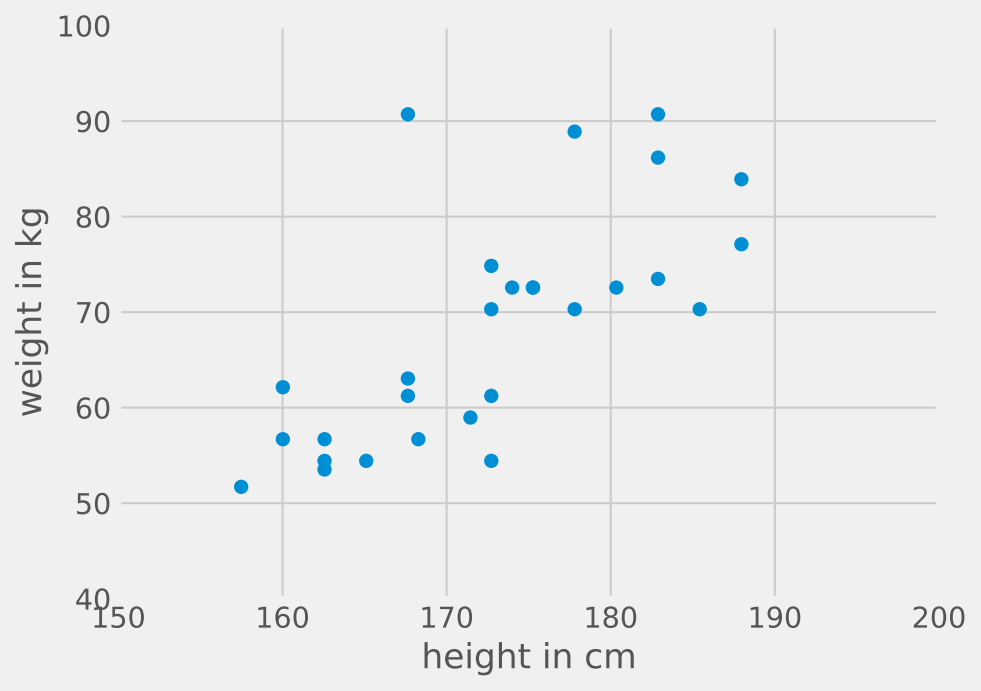
01 - Linear Regression/Least Squares

François Pitié

Assistant Professor in Media Signal Processing
Department of Electronic & Electrical Engineering, Trinity College Dublin

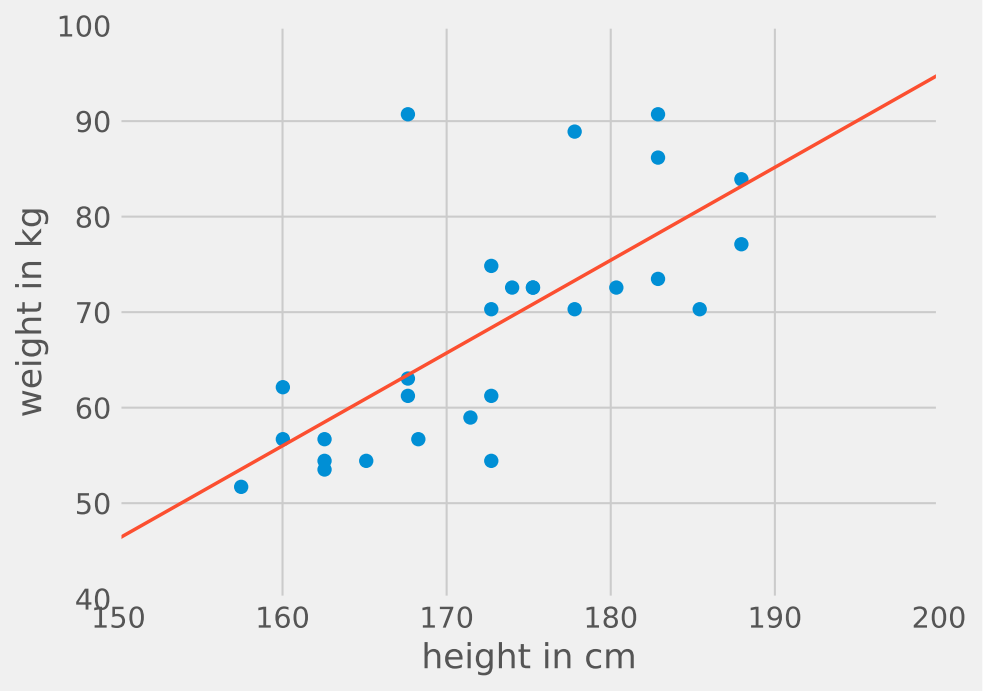
[4C16/5C16] Deep Learning and its Applications — 2023/2024

Linear Regression



We have collected some data.

Linear Regression



We are looking to infer a linear prediction:

$$\text{weight}(\text{kg}) = \text{height}(\text{cm}) \times 0.972 - 99.5$$

Linear Regression

The **input** is a feature vector $(x_1, \cdots, x_p)$.

The **output** is a scalar y. (You can easily generalise to a vector of output values by splitting the problem into as many sub-problems.)

Our **model** links the output y to the input feature vector $(x_1, \cdots, x_p)$ with a linear relationship:

$$y = w_0 + w_1 x_1 + w_2 x_2 + w_3 x_3 + \cdots + w_p x_p$$

Linear Regression

In practice, you have n observations, for which you have extracted p features:

$$
\begin{aligned}
y_1 &= w_0 + w_1 x_{11} + w_2 x_{12} + w_3 x_{13} + \cdots + w_p x_{1p} + \varepsilon_1 \\
y_2 &= w_0 + w_1 x_{21} + w_2 x_{22} + w_3 x_{23} + \cdots + w_p x_{1p} + \varepsilon_2 \\
y_3 &= w_0 + w_1 x_{31} + w_2 x_{32} + w_3 x_{33} + \cdots + w_p x_{3p} + \varepsilon_3 \\
&\vdots \\
y_n &= w_0 + w_1 x_{n1} + w_2 x_{n2} + w_3 x_{n3} + \cdots + w_p x_{np} + \varepsilon_n
\end{aligned}
$$

As the model can't explain everything we introduce an error term ε.

Linear Regression

We want to find the weight $w_0, w_1, \cdots, w_p$ that minimises the error.

At this point, the error $(\varepsilon_i)_{1 \leq i \leq n}$ is a vector of n separate terms. Since we can't minimise a vector, we need to aggregate the values into a single scalar that be used for comparison.

In linear regression, we choose to combine the error terms using the mean squared error (MSE):

$$E = \frac{1}{n} \sum_{i=1}^{n} \varepsilon_i^2 = \frac{1}{n} \sum_{i=1}^{n} (w_0 + w_1 x_{i1} + \cdots + w_p x_{ip} - y_i)^2$$

The choice of the mean squared error is a fundamental aspect of linear regression. Other error metrics are possible (eg. mean absolute difference) but they lead to very different mathematics.

Derivation Calculus

Let's derive the values $w_0, w_1, \cdots, w_p$ that minimise the mean squared error function $E(w_0, w_1, \cdots, w_p)$.

Derivation Calculus

$$E(w_0, \cdots, w_p) = \frac{1}{n} \sum_{i=1}^{n} (w_0 + w_1 x_{i1} + \cdots + w_p x_{ip} - y_i)^2$$

At the minimum of E, we have $\frac{\partial E}{\partial w_0} = \cdots = \frac{\partial E}{\partial w_p} = 0$:

$$\frac{\partial E}{\partial w_0}(w_0, \cdots, w_p) = \frac{2}{n} \sum_{i=1}^{n} (w_0 + w_1 x_{i1} + \cdots + w_p x_{ip} - y_i) = 0$$

$$\frac{\partial E}{\partial w_1}(w_0, \cdots, w_p) = \frac{2}{n} \sum_{i=1}^{n} x_{i1} (w_0 + w_1 x_{i1} + \cdots + w_p x_{ip} - y_i) = 0$$

$$\vdots$$

$$\frac{\partial E}{\partial w_p}(w_0, \cdots, w_p) = \frac{2}{n} \sum_{i=1}^{n} x_{ip} (w_0 + w_1 x_{i1} + \cdots + w_p x_{ip} - y_i) = 0$$

Derivation Calculus

Rearranging terms and dividing by $2/n$:

$$
\begin{aligned}
w_0 \sum_{i=1}^{n} 1 \quad &+ w_1 \sum_{i=1}^{n} x_{i1} \quad &&+ \cdots + w_p \sum_{i=1}^{n} x_{ip} \quad &&= \sum_{i=1}^{n} y_i \\
w_0 \sum_{i=1}^{n} x_{i1} &+ w_1 \sum_{i=1}^{n} x_{i1}^2 \quad &&+ \cdots + w_p \sum_{i=1}^{n} x_{i1} x_{ip} &&= \sum_{i=1}^{n} x_{i1} y_i \\
&\vdots \qquad\qquad &&\vdots \quad \vdots \qquad\qquad &&\vdots \\
w_0 \sum_{i=1}^{n} x_{ip} &+ w_1 \sum_{i=1}^{n} x_{ip} x_{i1} &&+ \cdots + w_p \sum_{i=1}^{n} x_{ip}^2 \quad &&= \sum_{i=1}^{n} x_{ip} y_i
\end{aligned}
$$

This gives us a linear system of $p+1$ equations, which can be solved efficiently using linear solvers.

Matrix Calculus

We are now going to derive the same equations using matrix notations. It is useful to know how to do this without having to come back to these sums and systems of equations.

Notations

By convention, we will write a scalar as x, a vector as $\mathbf{x}$ and a matrix as $\mathbf{X}$. We denote:

$$\mathbf{y} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix}, \ \mathbf{X} = \begin{pmatrix} 1 & x_{11} & \cdots & x_{1p} \\ 1 & x_{21} & \cdots & x_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_{n1} & \cdots & x_{np} \end{pmatrix}, \ \mathbf{w} = \begin{pmatrix} w_0 \\ w_1 \\ w_2 \\ \vdots \\ w_p \end{pmatrix}, \ \boldsymbol{\varepsilon} = \begin{pmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \vdots \\ \varepsilon_n \end{pmatrix}$$

The linear model then becomes:

$$\mathbf{y} = \mathbf{X}\mathbf{w} + \boldsymbol{\varepsilon}$$

The matrix $\mathbf{X}$, which stacks all the observations, is also called the Design Matrix.

Matrix Notations

Let us pause here for a second and come back to our model definition:

$$y = w_0 + w_1 x_1 + \cdots + w_p x_p$$

It strikes us that w_0 is a bit different from the other weights and it is no surprise that this weight w_0 is been given a special name in the deep learning community: **the bias**.

Matrix Notations

In the rest of the course, we will consider that the bias w_0 corresponds to the weight of a feature which turns out to have value 1. Thus we'll write the actual feature vector as:

$$\mathbf{x}_i = \begin{pmatrix} 1 \\ x_{i1} \\ \vdots \\ x_{ip} \end{pmatrix}$$

The linear model can then be written as: $y_i = \mathbf{x}_i^\top \mathbf{w}$

and the Design matrix as:

$$\mathbf{X} = \begin{pmatrix} \mathbf{x}_1^\top \\ \vdots \\ \mathbf{x}_n^\top \end{pmatrix}$$

Matrix Notations

Coming back to our problem, the mean squared error can be written in Matrix notations as:

$$
\begin{aligned}
E(\mathbf{w}) &= \frac{1}{n}\sum_{i=1}^{n}\varepsilon_i^2 = \frac{1}{n}\boldsymbol{\varepsilon}^\top\boldsymbol{\varepsilon} = \frac{1}{n}\|\boldsymbol{\varepsilon}\|^2 \\
&= \frac{1}{n}\left(\mathbf{X}\mathbf{w}-\mathbf{y}\right)^\top\left(\mathbf{X}\mathbf{w}-\mathbf{y}\right) \\
&= \frac{1}{n}\left(\mathbf{w}^\top\mathbf{X}^\top\mathbf{X}\mathbf{w}+\mathbf{y}^\top\mathbf{y}-2\mathbf{w}^\top\mathbf{X}^\top\mathbf{y}\right)
\end{aligned}
$$

At the minimum of $E(\mathbf{w})$, we have

$$
\frac{\partial E}{\partial \mathbf{w}} = \left(\frac{\partial E}{\partial w_0}, \cdots, \frac{\partial E}{\partial w_p}\right) = (0, \cdots, 0)
$$

$\frac{\partial E}{\partial \mathbf{w}}$ is the **gradient** of E and is often denoted as ∇E

Gradient Calculus

Knowing how to derive the gradient in matrix notations is very useful. Below is a list of useful gradient derivations.

We assume that the vectors $\mathbf{a}, \mathbf{b}$ and matrix $\mathbf{A}$ are independent of $\mathbf{w}$.

$$\frac{\partial \mathbf{a}^\top \mathbf{w}}{\partial \mathbf{w}} = \mathbf{a}$$

$$\frac{\partial \mathbf{b}^\top \mathbf{A} \mathbf{w}}{\partial \mathbf{w}} = \mathbf{A}^\top \mathbf{b}$$

$$\frac{\partial \mathbf{w}^\top \mathbf{A} \mathbf{w}}{\partial \mathbf{w}} = (\mathbf{A} + \mathbf{A}^\top)\mathbf{w} \qquad \text{(or } 2\mathbf{A}\mathbf{w} \text{ if } A \text{ symmetric)}$$

$$\frac{\partial \mathbf{w}^\top \mathbf{w}}{\partial \mathbf{w}} = 2\mathbf{w}$$

$$\frac{\partial\, \mathbf{a}^\top \mathbf{w} \mathbf{w}^\top \mathbf{b}}{\partial\, \mathbf{w}} = (\mathbf{a}\mathbf{b}^\top + \mathbf{b}\mathbf{a}^\top)\mathbf{w}$$

Exercise:

compute the gradient $\frac{\partial E(\mathbf{w})}{\partial \mathbf{w}}$ for

$$E(\mathbf{w}) = (\mathbf{w} - \mathbf{B}\mathbf{w})^\top \mathbf{A}(\mathbf{w} - \mathbf{a})$$

We have no assumptions about matrices $\mathbf{A}$ and $\mathbf{B}$.

Matrix Derivations

Let's come back to our problem:

$$\frac{\partial E}{\partial \mathbf{w}} = \frac{1}{n}\frac{\partial}{\partial \mathbf{w}}\left(\mathbf{w}^\top \mathbf{X}^\top \mathbf{X}\mathbf{w} + \mathbf{y}^\top \mathbf{y} - 2\mathbf{w}^\top \mathbf{X}^\top \mathbf{y}\right)$$

Apply the previous slide formula for each of the terms:

$$\frac{\partial}{\partial \mathbf{w}}\left(\mathbf{w}^\top \mathbf{X}^\top \mathbf{X}\mathbf{w}\right) = 2\mathbf{X}^\top \mathbf{X}\mathbf{w}$$

$$\frac{\partial}{\partial \mathbf{w}}\left(\mathbf{y}^\top \mathbf{y}\right) = 0$$

$$\frac{\partial}{\partial \mathbf{w}}\left(\mathbf{w}^\top \mathbf{X}^\top \mathbf{y}\right) = \mathbf{X}^\top \mathbf{y}$$

Thus

$$\frac{\partial E}{\partial \mathbf{w}} = \frac{2}{n}\mathbf{X}^\top \mathbf{X}\mathbf{w} - \frac{2}{n}\mathbf{X}^\top \mathbf{y} = 0$$

which can be simplified as follows:

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

Matrix Derivations

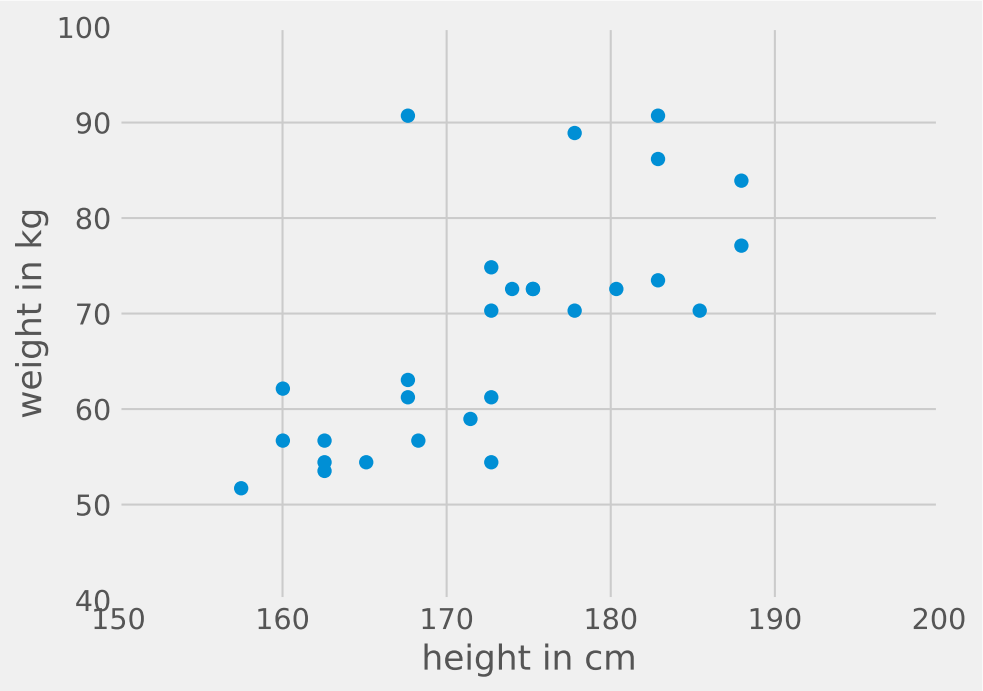
This is called the normal equation:

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

and it is the same as our linear system in slide 9.

Curve Fitting

Let’s come back to our original problem ...



Curve Fitting

...and derive the normal equations using matrix notations.

The model is affine $y = w_0 + w_1 x$.

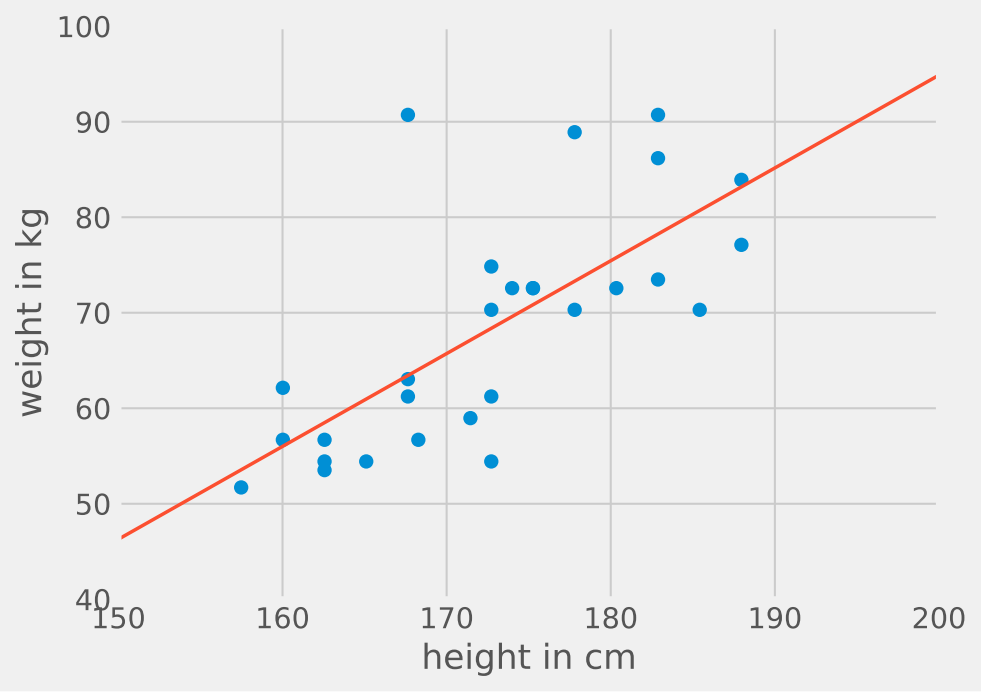
The design matrix that stacks all features is thus $\mathbf{X} = \begin{pmatrix} 1 & x_1 \\ 1 & x_2 \\ \vdots & \vdots \\ 1 & x_n \end{pmatrix}$

and the matrices of the normal equations are:

$$\mathbf{X}^\top \mathbf{X} = \begin{pmatrix} \sum_{i=1}^n 1 & \sum_{i=1}^n x_i \\ \sum_{i=1}^n x_i & \sum_{i=1}^n x_i^2 \end{pmatrix}, \quad \mathbf{X}^\top \mathbf{y} = \begin{pmatrix} \sum_{i=1}^n y_i \\ \sum_{i=1}^n x_i y_i \end{pmatrix}$$

The LS estimate is then:

$$\hat{\mathbf{w}} = \left(\mathbf{X}^\top \mathbf{X}\right)^{-1} \mathbf{X}^\top \mathbf{y} = \begin{pmatrix} \sum_{i=1}^n 1 & \sum_{i=1}^n x_i \\ \sum_{i=1}^n x_i & \sum_{i=1}^n x_i^2 \end{pmatrix}^{-1} \begin{pmatrix} \sum_{i=1}^n y_i \\ \sum_{i=1}^n x_i y_i \end{pmatrix}$$



We find numerically that $\hat{\mathbf{w}} = \begin{pmatrix} -99.5 \\ 0.972 \end{pmatrix}$, which corresponds to a least square linear model as follows:

$$\text{weight} = \text{height} \times 0.97299.5$$

Linear Models

Although the model is linear, it doesn't mean that we can only fit linear or affine curves. Consider the following model:

$$y = w_0 + w_1 x + w_2 x^2 + w_3 x^3$$

This is still a "linear" model because y is a linear combination of the input 1, x, x^2 and x^3. What we have done is changed the input feature $\mathbf{x} = [1, x]$ to $\mathbf{x} = [1, x, x^2, x^3]$. Here is another example:

$$y = w_0 + w_1 \cos(2\pi x) + w_2 \sin(2\pi x)$$

is also a linear model in the parameters w_0, w_1, w_2 with input vector $\mathbf{x} = [1, \cos(2\pi x), \sin(2\pi x)]$. Note that

$$y = w_0^2 + x$$

is not linear in the parameters because w_0^2 is not linear in w_0.

Linear Models

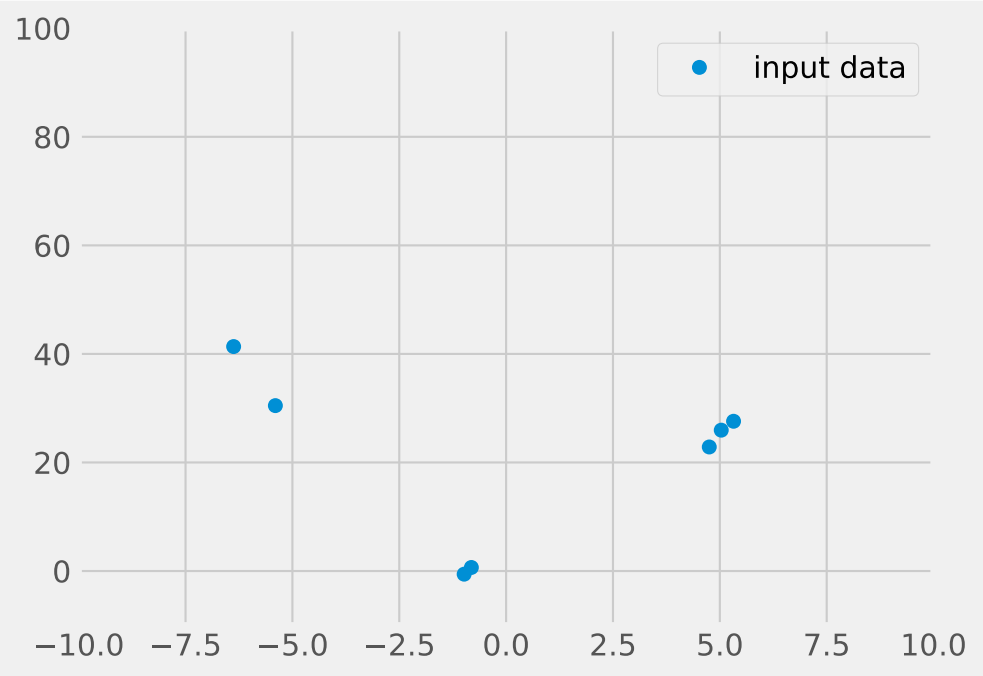
A model will be linear if you can write the model as follows:

$$y = f(\mathbf{x}, \mathbf{w}) = \sum_{i=0}^{p} w_i f_i(\mathbf{x})$$

where the functions f_i are independent of $\mathbf{w}$.

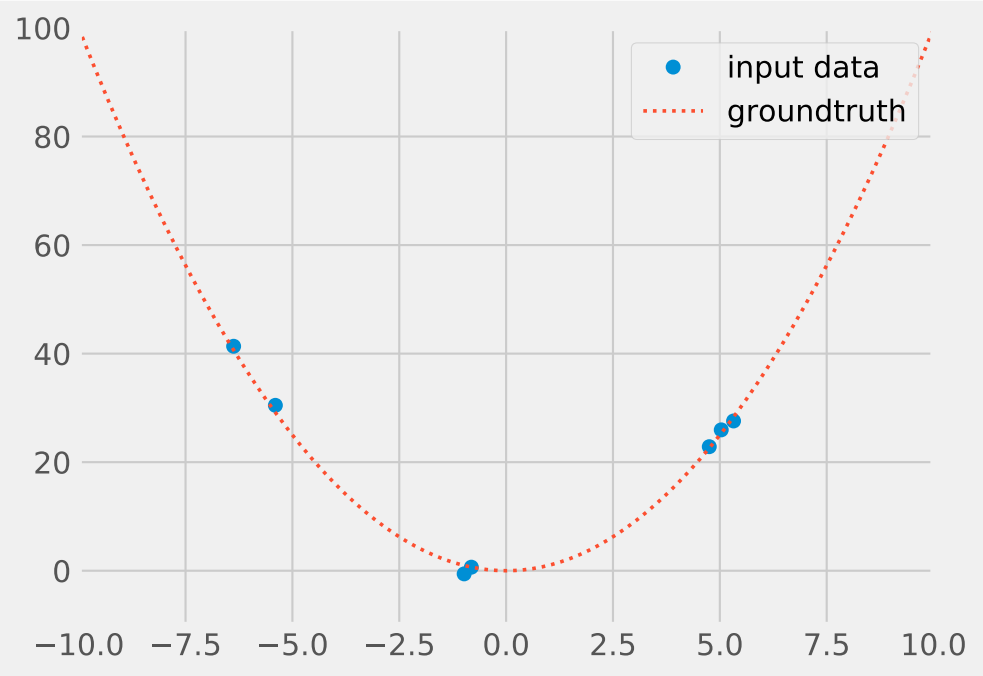
Curve Fitting

Here is an example of using Least Square for polynomial fitting.



Curve Fitting

The true model is of the form: $y = w_0 + w_1 x + w_2 x^2$



Curve Fitting

Let's derive the normal equations using matrix notations.

The model is $y = w_0 + w_1 x + w_2 x^2$, thus the features are

$$\mathbf{X} = \begin{pmatrix} 1 & x_1 & x_1^2 \\ 1 & x_2 & x_2^2 \\ \vdots & \vdots & \vdots \\ 1 & x_n & x_n^2 \end{pmatrix}$$

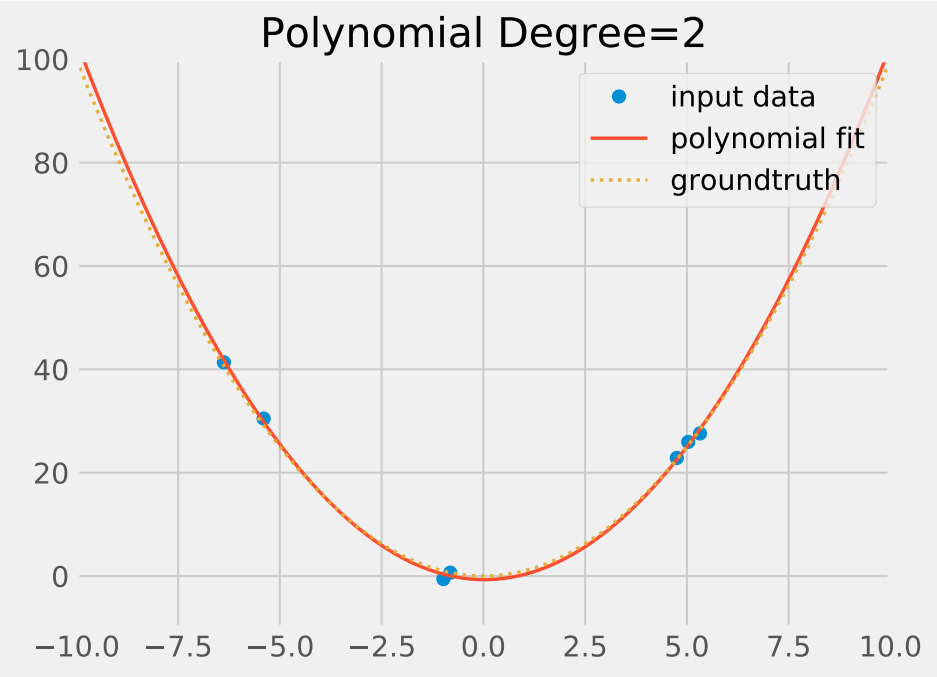
$$\mathbf{X}^\top \mathbf{X} = \begin{pmatrix} \sum_{i=1}^n 1 & \sum_{i=1}^n x_i & \sum_{i=1}^n x_i^2 \\ \sum_{i=1}^n x_i & \sum_{i=1}^n x_i^2 & \sum_{i=1}^n x_i^3 \\ \sum_{i=1}^n x_i^2 & \sum_{i=1}^n x_i^3 & \sum_{i=1}^n x_i^4 \end{pmatrix}, \quad \mathbf{X}^\top \mathbf{y} = \begin{pmatrix} \sum_{i=1}^n y_i \\ \sum_{i=1}^n x_i y_i \\ \sum_{i=1}^n x_i^2 y_i \end{pmatrix}$$

The LS estimate is then:

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

Curve Fitting

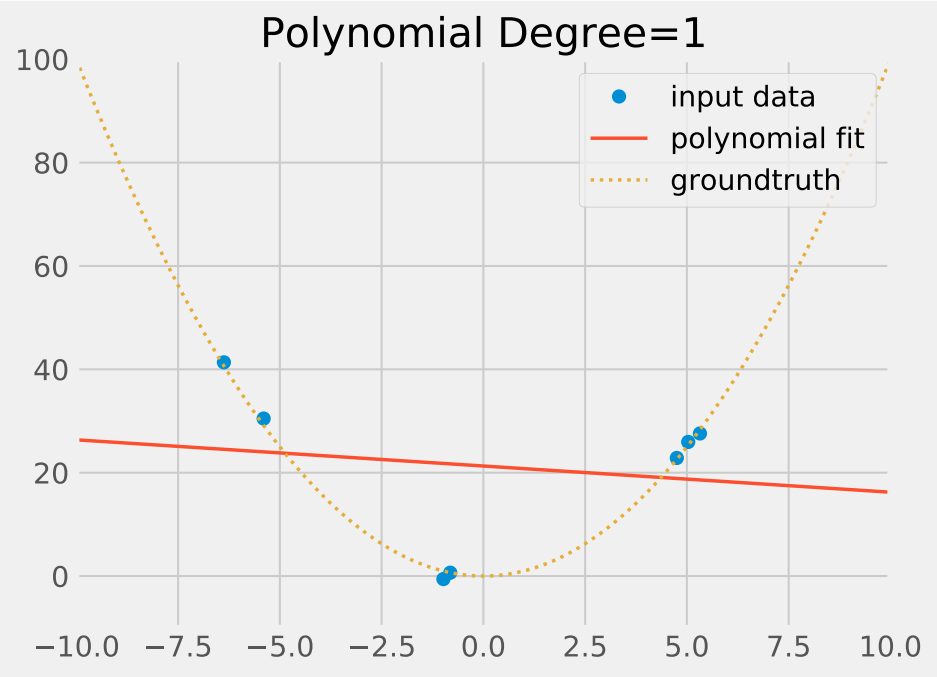
This is what the LS estimate looks like:



MSE: 4.38e-01

Underfitting

Let's see what happens when you try to fit the data with a lower model order: $y = w_0 + w_1 x$



MSE: 2.02e+02

Underfitting

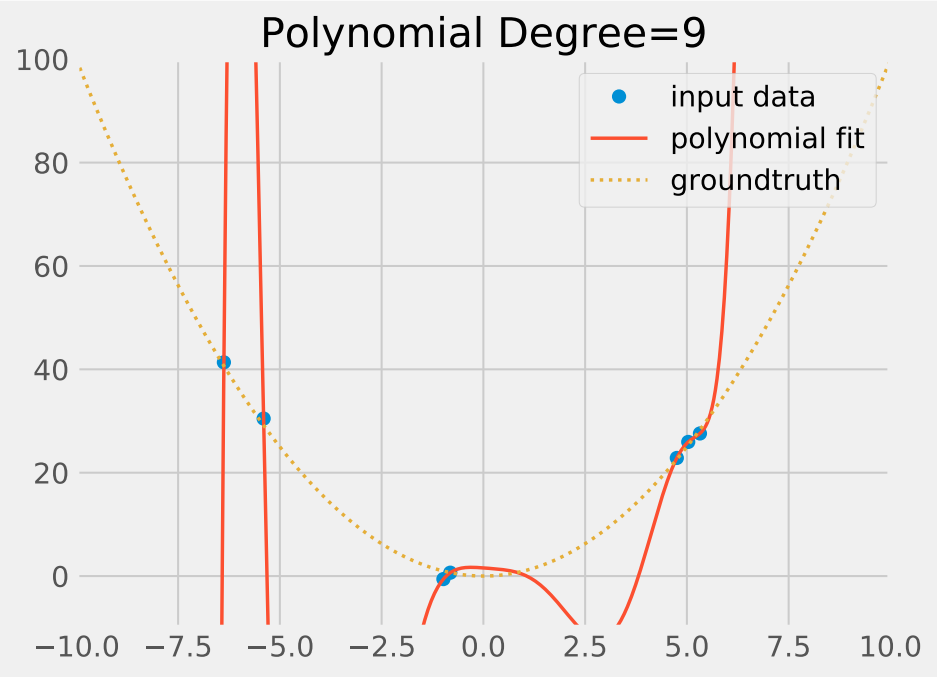
This problem is called underfitting. This is a frequent problem in machine learning.

How do you know that your are underfitting?

You know that you are underfitting when the error cannot get low enough.

Overfitting

Let's now try a higher model order: $y = w_0 + w_1 x + \cdots + w_9 x^9$



MSE: 7.59e-06

Overfitting

Although the error on the observed data is perfect (MSE=7.59e-06), it is clear that the predicted model is largely incorrect in-between the observed data.

This problem is called overfitting and it is also a fundamental problem in machine learning.

It boils down to this: given enough parameters your model will fit pretty much anything. But that doesn't mean your model can generalise well to any data outside the data used for training.

How to know if you are overfitting?

You know that you are overfitting when the error is very low on the data used for training but quite high on newly predicted data.

How to combat overfitting?

The first thing you can check is if the chosen model is too complex for the data. Thus use a simpler model and make sure you are not under-fitting.

eg. you are fitting a polynomial of order 9 but the true model is in fact of order 2

How to combat overfitting?

Most of the time the model is fine but you simply don't have enough observations to fit our model.

The cure is then to get more data.

eg. you only use 5 points to fit a polynomial of order 9, you need more data.

How to combat overfitting?

Using more data is always a good idea. It allows you to use models that are more complex than necessary. If some features are not useful, you can expect that the corresponding estimated weights w_i will shrink to zero.

eg. It is OK to fit a polynomial of order 9 when the underlying model is actually of order 2. Just make sure you have plenty of data.

How to combat overfitting?

But what if you can't get enough data?

Get more.

How to combat overfitting?

But what if <u>really</u> can't?

One alternative solution to more data is to use **regularisation**.

Tikhonov Regularisation

In Least Squares, a natural regularisation technique is called the **Tikhonov regularisation**.

Instead of minimising $\|\varepsilon\|^2 = \|\mathbf{X}\mathbf{w} - \mathbf{y}\|^2$, we minimise a slightly modified expression:

$$E(\mathbf{w}) = \|\mathbf{X}\mathbf{w} - \mathbf{y}\|^2 + \alpha\|\mathbf{w}\|^2$$

This extra term improves the conditioning of the problem (ie. making the matrix $\mathbf{X}^\top\mathbf{X}$ invertible) whilst still having a direct solution:

$$\hat{\mathbf{w}} = (\mathbf{X}^\top\mathbf{X} + \alpha\mathbf{I})^{-1}\mathbf{X}^\top\mathbf{y}$$

where $\mathbf{I}$ is the identity matrix (with zeros everywhere and ones on the diagonal).

Regularisation

Basically the effect of the Tikhonov regularization is to penalise the parameters values $\mathbf{w}$ when they are far away from 0. It thus acts as a bias that pulls the estimation of $\mathbf{w}$ slightly towards 0.

The motivation is that, given no other information, it is more likely that the weights $\mathbf{w}$ are small than high.

eg. it is more likely to have

$$\text{weight} = \text{height} \times 0.97299.5$$

than

$$\text{weight} = \text{height} \times 10^{10} 10^{20}$$

Thus, as a default, we should favour weights $\mathbf{w}$ that are closer to zero.

Regularisation

Regularisation is often a necessary evil. It allows you to avoid gross errors when predicting samples that are far outside the range of the training data.

But this comes at the cost of biasing the estimation.

Thus in practice you want to avoid it.

A good way to avoid regularisation is to get enough data so that $\mathbf{X}^\top \mathbf{X}$ becomes comfortably invertible.

So get plenty data!

Maximum Likelihood

Very early on, Gauss connected Least squares with the principles of probability and to the Gaussian distribution.

Maximum Likelihood

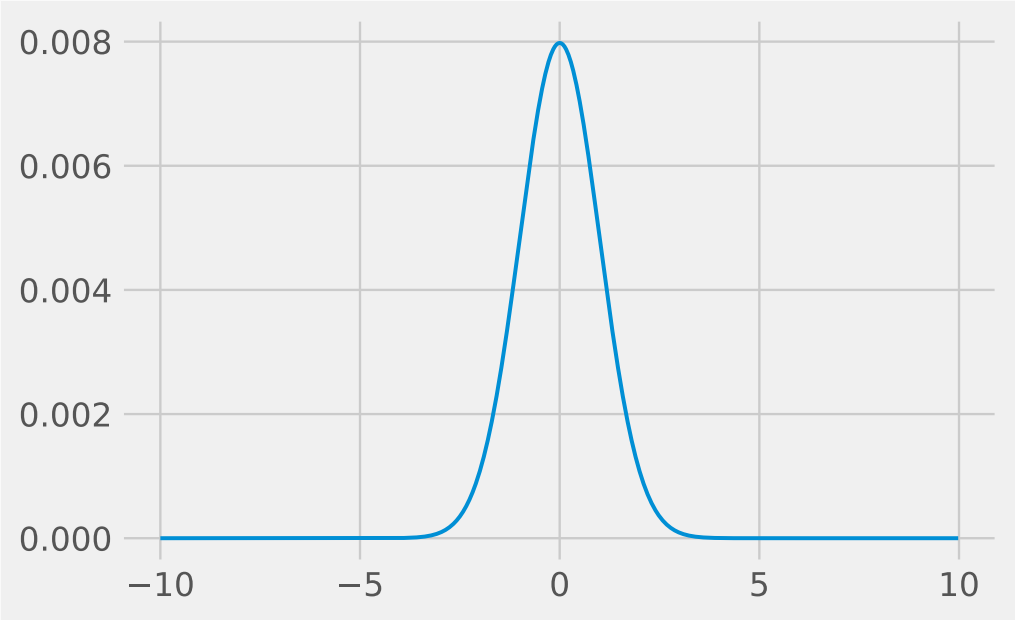
Recall that the linear model is:

$$\mathbf{y} = \mathbf{X}\mathbf{w} + \boldsymbol{\varepsilon}$$

Let's give a probabilistic view on this by assuming that the error $\boldsymbol{\varepsilon}$ follows a Gaussian distribution:

$$\boldsymbol{\varepsilon} \sim \mathcal{N}(0, \sigma^2)$$

$$p(\varepsilon) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{\varepsilon^2}{2\sigma^2}}$$



Maximum Likelihood

The likelihood to have y_i given $\mathbf{x}_i$ is

$$p(y_i|\mathbf{x}_i, \mathbf{w}) = p(\varepsilon_i = \mathbf{x}_i^\top \mathbf{w} - y_i) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(\mathbf{x}_i^\top \mathbf{w} - y_i)^2}{2\sigma^2}\right)$$

The likelihood to have all outputs $\mathbf{y}$ given all data $\mathbf{X}$ is given by

$$\begin{aligned} p(\mathbf{y}|\mathbf{X}, \mathbf{w}) &= \prod_{i=1}^{n} p(\varepsilon_i) \\ &= \left(\frac{1}{\sqrt{2\pi\sigma^2}}\right)^n \exp\left(-\sum_{i=1}^{n} \frac{\left(\mathbf{x}_i^\top \mathbf{w} - y_i\right)^2}{2\sigma^2}\right) \end{aligned}$$

Maximum Likelihood

We seek to find the maximum likelihood estimate of $\mathbf{w}$. That is, finding $\mathbf{w}$ that maximises the likelihood $p(\mathbf{y}|\mathbf{X}, \mathbf{w})$:

$$\hat{\mathbf{w}}_{ML} = \arg\max_{\mathbf{w}} p(\mathbf{y}|\mathbf{X}, \mathbf{w})$$

A more practical, but equivalent, approach is to minimise the negative log likelihood:

$$\begin{aligned}
\hat{\mathbf{w}}_{ML} &= \arg\min_{\mathbf{w}} -\log\left(p(\mathbf{y}|\mathbf{X}, \mathbf{w})\right) \\
&= \arg\min_{\mathbf{w}} \frac{1}{2\sigma^2} \sum_{i=1}^{n} \left(\mathbf{x}_i^\top \mathbf{w} - y_i\right)^2 - n\log\left(\sqrt{2\pi\sigma^2}\right) \\
&= \arg\min_{\mathbf{w}} \sum_{i=1}^{n} \left(\mathbf{x}_i^\top \mathbf{w} - y_i\right)^2
\end{aligned}$$

Maximum Likelihood

Thus we've shown that the Least Square estimate is in fact the Maximum Likelihood solution if the error is assumed to be Gaussian.

Historical Notes

Origins of Least Squares

The least-squares method has its origins in the methods of calculating orbits of celestial bodies. It is often credited to Carl Friedrich Gauss (1809) but it was first published by Adrien-Marie Legendre in 1805. The priority dispute comes from Gauss's claim to have used least squares since 1795.

APPENDICE.

Sur la Méthode des moindres quarrés.

Dans la plupart des questions où il s'agit de tirer des mesures données par l'observation, les résultats les plus exacts qu'elles peuvent offrir, on est presque toujours conduit à un système d'équations de la forme

$$E = a + bx + cy + fz + \&c.$$

dans lesquelles a, b, c, f, &c. sont des coëfficiens connus, qui varient d'une équation à l'autre, et x, y, z, &c. sont des inconnues qu'il faut déterminer par la condition que la valeur de E se réduise, pour chaque équation, à une quantité ou nulle ou très-petite.

Legendre (1805), *Nouvelles méthodes pour la détermination des orbites des comètes.*

Origins of Regression

The term **Regression** comes from the publication by Francis Galton *Regression towards mediocrity in hereditary stature (1886)*.

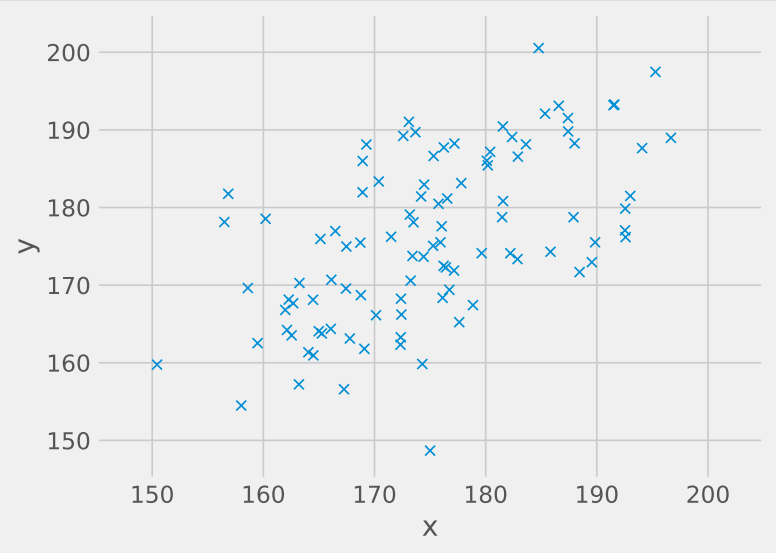
Galton was comparing the distribution of heights from parents and their offsprings. He applied Least Squares and observed that his linear fit predicted that parents who are at the tails of the height distribution have offsprings that are closer to the mean of the distribution, eg. taller than average parents are predicted to have shorter offsprings.

Hence the expression **"regression"** towards the mean.

In fact this is a fallacy and a misuse of Least Squares.

Origins of Regression

This is what the scatter plot of the parent's height x vs. the offspring's height y looks like:



The problem is that both measured heights x and y are noisy measurements of some true underlying "height genes" u and v.

The underlying linear model $v = w_1 u + w_0$ becomes:

$$y + \epsilon = w_1 (x + \nu) + w_0$$

where ϵ and ν are our noise variables.

What is the impact of having noisy features $x+\nu$ on the normal equations?

$$\mathbf{X}'^\top \mathbf{X}' = \begin{pmatrix} \sum_{i=1}^n 1 & \sum_{i=1}^n x_i + \nu_i \\ \sum_{i=1}^n x_i + \nu_i & \sum_{i=1}^n (x_i + \nu_i)^2 \end{pmatrix} \quad , \quad \mathbf{X}'^\top \mathbf{y} = \begin{pmatrix} \sum_{i=1}^n y_i \\ \sum_{i=1}^n (x_i + \nu_i) y_i \end{pmatrix}$$

assuming that ν is independent of x, we have:
$\lim_{n\to\infty} \frac{1}{n} \sum_{i=1}^n \nu_i = 0$, $\lim_{n\to\infty} \frac{1}{n} \sum_{i=1}^n \nu_i x_i = 0$, $\lim_{n\to\infty} \frac{1}{n} \sum_{i=1}^n \nu_i y_i = 0$

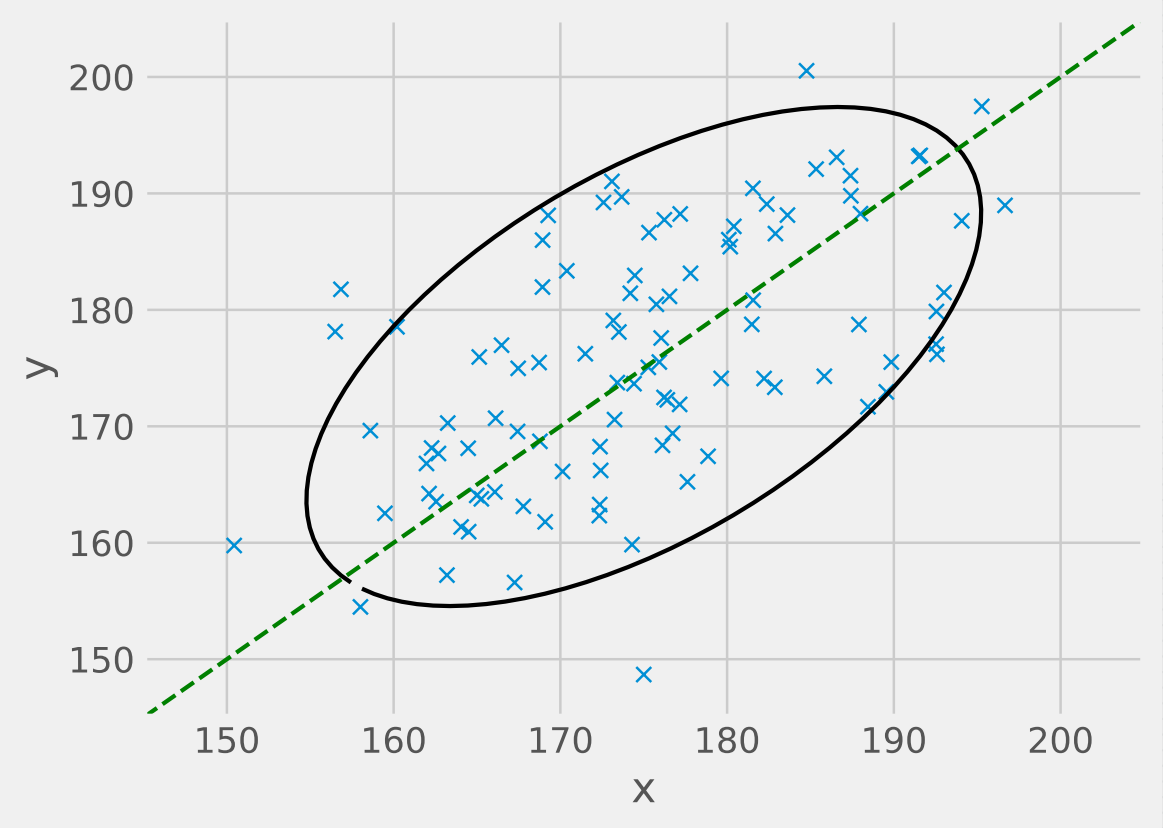
Thus

$$\mathbf{w} \approx \left(\mathbf{X}^\top \mathbf{X} + n \begin{pmatrix} 0 & 0 \\ 0 & \sigma_\nu^2 \end{pmatrix} \right)^{-1} \mathbf{X}^\top \mathbf{y}$$

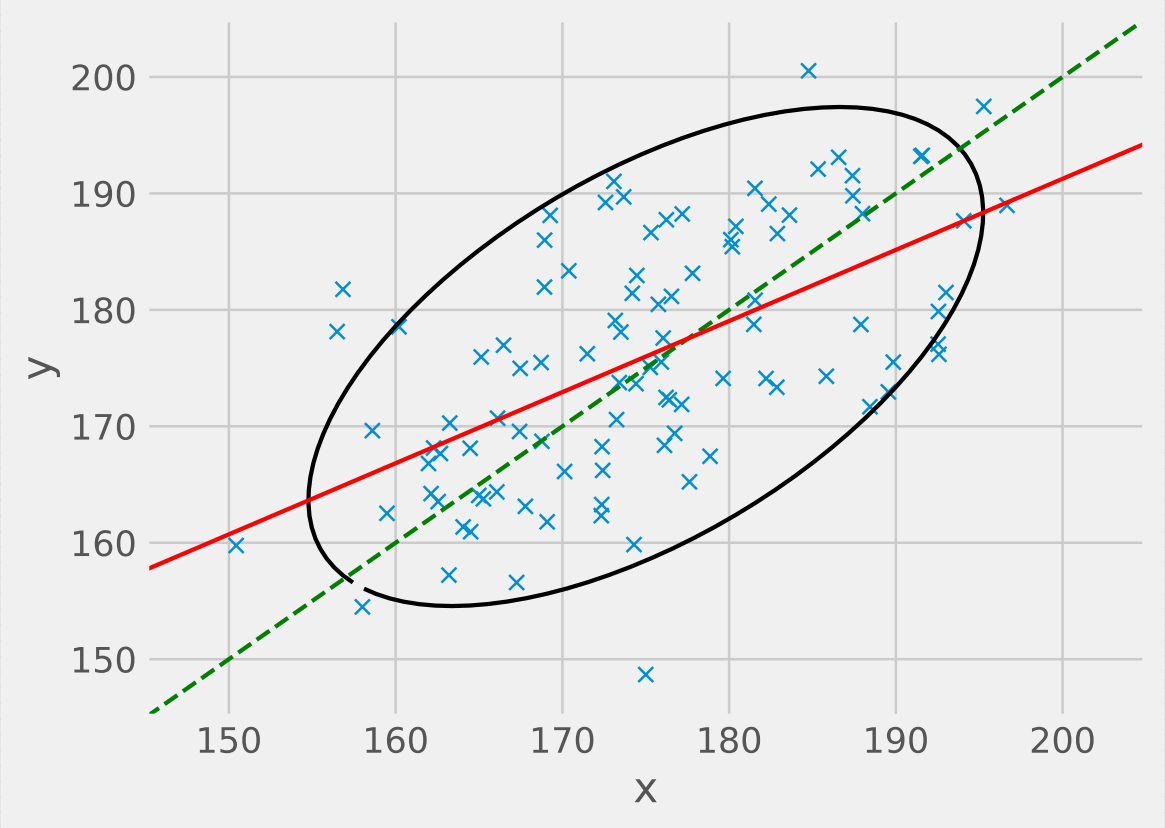
with $\sigma_\nu^2 = \frac{1}{n} \sum_i \nu_i^2$ the noise variance.

This is similar to what we've seen with the Tikhonov regularisation. The noise ν biases w_1 towards 0.

This is the Ground-Truth of the linear model:



but this is what you get with a LS estimate:



Conclusion: if your features are noisy, then LS will bias your parameter estimation towards 0.

Always check your model. LS assumes $y = \mathbf{x}^\top \mathbf{w} + \epsilon$ with ϵ Gaussian. If your model is $y = (\mathbf{x} + \nu)^\top \mathbf{w} + \epsilon$, it is not the same.

Take Away

We start from a collection of n examples $(\mathbf{x}_i, y_i)_i$. Each of the examples was made up of a number p of features $\mathbf{x}_i = (x_1, \cdots, x_p)$.

We assume that the output can be predicted by a linear model: $y_i = \mathbf{x}_i^\top \mathbf{w} + \varepsilon_i$, with some error ε_i.

We combine all the errors term into a loss function, which is set to be the mean squared error of ε.

The parameters $\hat{\mathbf{w}}$ that minimise the loss function can be derived with the normal equations.

Least square estimation is equivalent is the maximum likelihood solution when we assume that ε follows a Gaussian distribution.

Two issues arise when solving for the LS estimate: underfitting and overfitting. One way to combat overfitting is to use more data and/or to use regularisation.