

Linear Regression

Prof. Alessandro Lucantonio

Aarhus University - Department of Mechanical and Production Engineering

?/?/2023

Weight-Height example

Dataset: heights and weights of different people.

Task: build a model that predict the height given the weight.

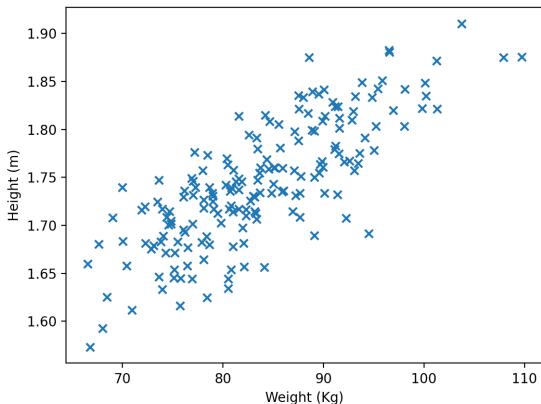


Figure: Data plot

A solution - Linear regression model

Some remarks on data.

- ▶ Regression problem (continuous output).
- ▶ Data with different order of magnitude.

A possible solution to this problem is represented by **linear regression** (LR).

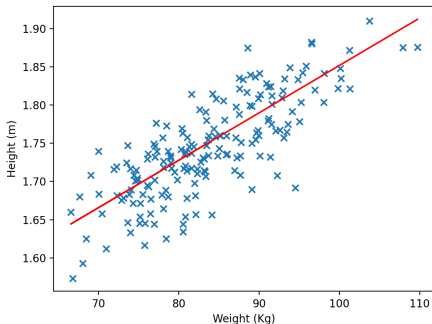


Figure: Trained model (in red)

General ingredients

Notation:

- ▶ x : a data sample.
- ▶ y : the data target corresponding to x
- ▶ N : number of data.

Model/hypothesis: $h_{\mathbf{w}}(x) = w_1x + w_0$, where $\mathbf{w} = [w_0, w_1]$ is the vector of parameter that has to be learned.

In our example, x is the weight of a single sample and $h_{\mathbf{w}}(x)$ corresponds to the prediction of its height.

Usually the vector \mathbf{w} is called **weights vector** and the set $\mathcal{H} := \{h_{\mathbf{w}} | \mathbf{w} \in \mathbb{R}^2\}$ is called **hypothesis space**.

How to learn w from data?

Mean squared error (MSE)

Given a training sample x_i and a model $h_{\mathbf{w}}$ we can predict the target computing $h_{\mathbf{w}}(x_i)$. To evaluate how good is the prediction we compute the error $(h_{\mathbf{w}}(x_i) - y_i)^2$.

$(h_{\mathbf{w}}(x_i) - y_i)^2 \geq 0$ and $(h_{\mathbf{w}}(x_i) - y_i)^2 = 0$ if and only if $h_{\mathbf{w}}(x_i) = y_i$. The **mean squared error** (MSE) is:

$$E(\mathbf{w}) := \frac{1}{N} \sum_{i=1}^N (h_{\mathbf{w}}(x_i) - y_i)^2.$$

To find the best model we minimize the training error, hence in this case the MSE.

$$\mathbf{w} \in \arg \min_{\tilde{\mathbf{w}} \in \mathbb{R}^2} E(\tilde{\mathbf{w}}).$$

n -dimensional LR

Dataset samples.

Previous case: $x \in \mathbb{R}, y \in \mathbb{R}$.

Now: $\mathbf{x} \in \mathbb{R}^n, y \in \mathbb{R}$.

Notation: x_j^i is the j -th coordinate of the i -th sample.

Hypothesis.

Previous case:

$$h_w(x) = w_1 x + w_0,$$

where $w = [w_0, w_1]$.

Now:

$$\begin{aligned} h_{\mathbf{w}}(\mathbf{x}) &= w_n x_n + w_{n-1} x_{n-1} + \cdots + w_1 x_1 + w_0 \\ &= \sum_{i=0}^n w_i \tilde{x}_i = \mathbf{w}^T \tilde{\mathbf{x}}, \end{aligned}$$

where $\mathbf{w} = [w_0, \dots, w_n]$ and $\tilde{\mathbf{x}} = [1, x_1, \dots, x_n]$.

n-dimensional LR

MSE.

Previous case:

$$E(\mathbf{w}) = \frac{1}{N} \sum_{i=1}^N (h_{\mathbf{w}}(x_i) - y_i)^2.$$

Now:

$$\begin{aligned} E(\mathbf{w}) &= \frac{1}{N} \sum_{i=1}^N (h_w(\mathbf{x}^i) - y^i)^2 \\ &= \frac{1}{N} (\mathbf{X}\mathbf{w} - \mathbf{y})^T (\mathbf{X}\mathbf{w} - \mathbf{y}) \\ &= \frac{1}{N} \|\mathbf{X}\mathbf{w} - \mathbf{y}\|^2 \end{aligned}$$

where

$$\mathbf{X} := \begin{bmatrix} \tilde{\mathbf{x}}^1 \\ \vdots \\ \tilde{\mathbf{x}}^N \end{bmatrix} \quad \mathbf{y} = \begin{bmatrix} y_1 \\ \vdots \\ y_N \end{bmatrix}.$$

Spot the minimum - Gradient descent

How to find $\mathbf{w} \in \arg \min_{\tilde{\mathbf{w}} \in \mathbb{R}^2} E(\tilde{\mathbf{w}})$?

Main idea: the gradient of a scalar field represent geometrically the direction with maximum slope. Hence, following the opposite direction of the gradient lead us to get closer to the minimum of the function.

Formally:

- ▶ Start with a random \mathbf{w}^0 .
- ▶ For $j \geq 0$, update $\mathbf{w}^{j+1} := \mathbf{w}^j + \mathbf{d}^j$, where \mathbf{d}^j is such that

$$E(\mathbf{w}^{j+1}) \leq E(\mathbf{w}^j)$$

Gradient descent: $\mathbf{d}^j = -\alpha \nabla E(\mathbf{w}^j)$. α is called **learning rate**.

Gradient descent - 3D visualization

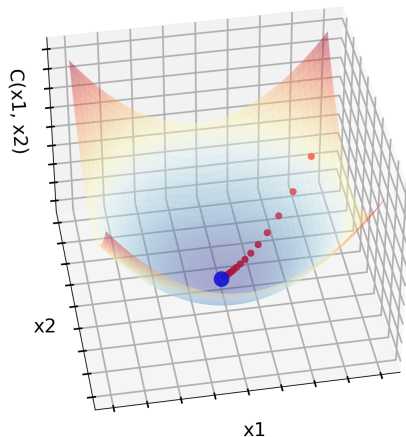


Figure: In blue the global minimum, in red the iteration points.

Gradient descent - 2D visualization

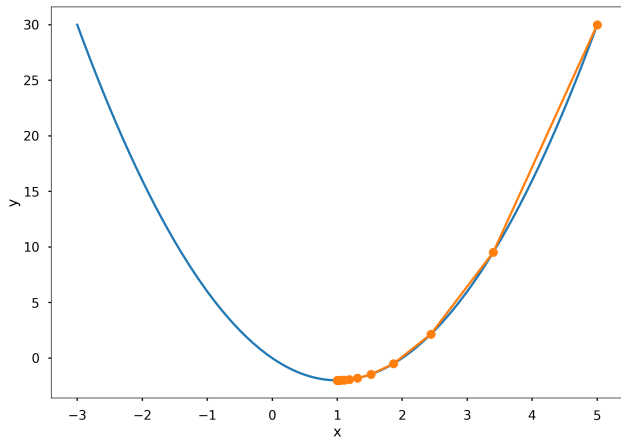


Figure: Learning rate = 0.1

Gradient descent - 2D visualization

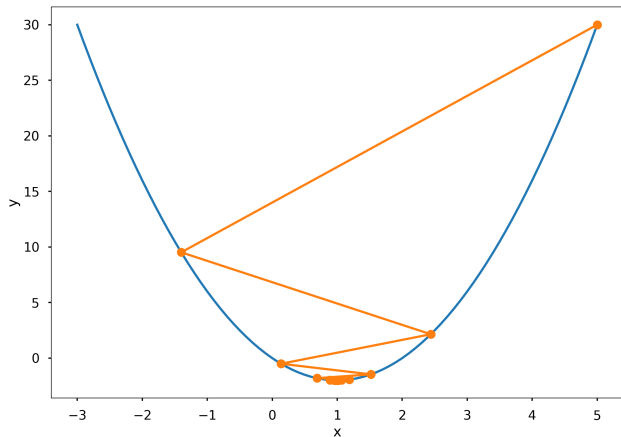


Figure: Learning rate = 0.4

Gradient descent - 2D visualization

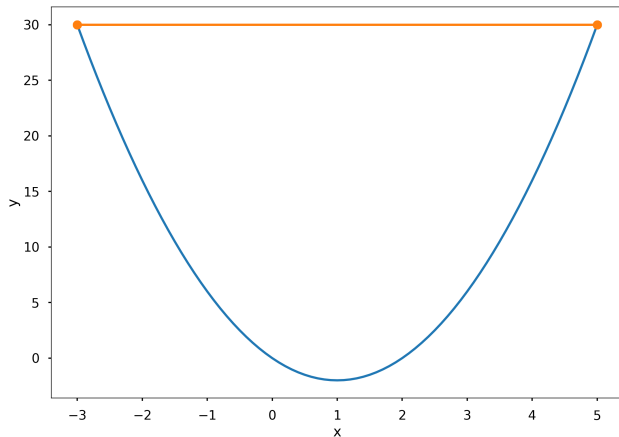


Figure: Learning rate = 0.5

Batch, SGD and Mini-Batch

Notation: $E(\mathbf{w}) = 1/N \sum_{i=1}^N E_i(\mathbf{w})$

Batch

- ▶ Start with a random \mathbf{w}^0 .
- ▶ For $j \geq 0$, update $\mathbf{w}^{j+1} := \mathbf{w}^j - \alpha \nabla E(\mathbf{w}^j)$.

Stochastic Gradient Descent (SGD or online)

- ▶ Start with a random \mathbf{w}^0 .
- ▶ For $j \geq 0$ and for each pattern $1 \leq i \leq N$ update $\mathbf{w}^{j+1} := \mathbf{w}^j - \alpha \nabla E_i(\mathbf{w}^j)$.

Mini-Batch. Fix an integer $1 \leq \text{mb} \leq N$ (mini-batch size).

- ▶ Start with a random \mathbf{w}^0 .
- ▶ For $j \geq 0$ and for each pattern $0 \leq i < \frac{N}{\text{mb}}$ update

$$\mathbf{w}^{j+1} := \mathbf{w}^j - \alpha \nabla \sum_{k=i \cdot \text{mb} + 1}^{(i+1) \cdot \text{mb}} E_k(\mathbf{w}^j)$$

Tips and Tricks - How to choose?

- ▶ Batch: usually more stable and provide a more accurate estimation of the gradient, but very slow.
- ▶ SGD: very fast, stochastic approximation of the gradient implies possible instability (Zig-zag effect)
- ▶ Mini-Batch: a trade-off (parallelism available).

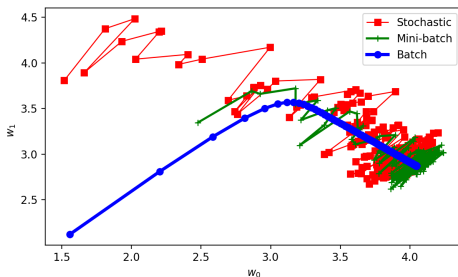


Figure: Batch vs SGD vs Mini-batch

Gradient descent and normal equation for LR

We have $E(\mathbf{w}) = \frac{1}{N} \|\mathbf{X}\mathbf{w} - \mathbf{y}\|^2$, hence

$$\nabla E(\mathbf{w}) = \frac{1}{N} \nabla (\|\mathbf{X}\mathbf{w} - \mathbf{y}\|^2) = \frac{2}{N} \mathbf{X}^T (\mathbf{X}\mathbf{w} - \mathbf{y})$$

Normal equation (\iff holds if $\mathbf{X}^T \mathbf{X}$ is invertible):

$$\begin{aligned} \nabla E(\mathbf{w}) = 0 &\iff \frac{2}{N} \mathbf{X}^T (\mathbf{X}\mathbf{w} - \mathbf{y}) = 0 \\ &\iff \mathbf{X}^T \mathbf{X} \mathbf{w} = \mathbf{X}^T \mathbf{y} \\ &\iff \mathbf{w} = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y} \end{aligned}$$

Gradient descent main iteration for LR:

$$\mathbf{w}^{j+1} := \mathbf{w}^j - \frac{2\alpha}{N} \mathbf{X}^T (\mathbf{X}\mathbf{w}^j - \mathbf{y})$$

Normal equation vs gradient descent

Normal equation:

- ▶ No hyperparameter (explicit solution).
- ▶ No need to iterate.
- ▶ $\mathcal{O}(n^3)$, hence slow when n is large.

Gradient descent:

- ▶ Need to choose the learning rate α .
- ▶ Needs many iterations.
- ▶ $\mathcal{O}(kn^2)$, hence fast when n is large.

Tips and Tricks - Standardization

General (not only for LR): features must be on a similar scale!

- ▶ Speed up the convergence of gradient descent.
- ▶ Try to have (on average) $-1 \leq x^i \leq 1$.

Common techniques:

- ▶ **Feature scaling.** Compute the max M and the min m data value. Then normalize each feature as follows

$$x_{\text{norm}}^i = \frac{x^i - m}{M - m}$$

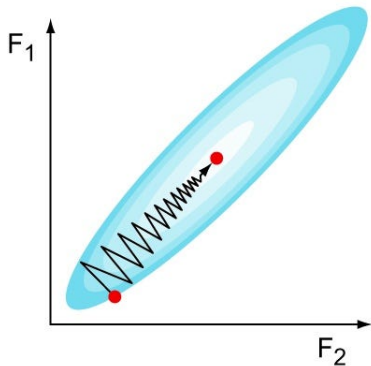
- ▶ **Mean normalization.** Compute mean μ and standard deviation σ of the data. Then normalize each feature as follows

$$x_{\text{norm}}^i = \frac{x^i - \mu}{\sigma}$$

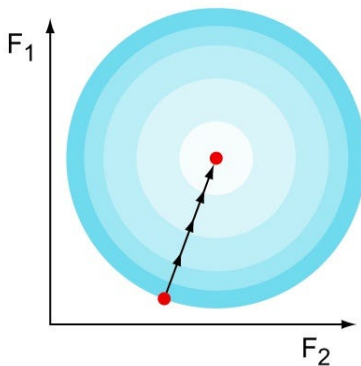
Tips and Tricks - Standardization

Gradient descent with and without feature scaling

Non-normalized features



Normalized features



Tips and Tricks - Invertibility of $X^T X$

What happens when $X^T X$ is not invertible?

Invertibility of $X^T X$ = column of X linearly independent
(preprocessing information)

If a column is linearly dependent to the other then a feature is correlated with others (redundant feature).

Solution: discard that feature. The information carried by that feature is contained in some of the others.

Polynomial regression (PR)

LR corresponds to linear hypothesis, i.e. of the form

$$h_{\mathbf{w}}(\mathbf{x}) = \mathbf{w}^T \mathbf{x}$$

PR corresponds to polynomial hypothesis, i.e. of the form

$$h_{\mathbf{w}}(\mathbf{x}) = \sum_{j=0}^n w_j x_j^j.$$

More in general: linear basis expansion (LBE)

$$h_{\mathbf{w}}(\mathbf{x}) = \sum_{j=0}^n w_j \phi_j(\mathbf{x}),$$

where $\phi_j : \mathbb{R}^n \rightarrow \mathbb{R}$.

Underfitting and overfitting - main intuition

Imagine that you have to prepare an exam.

Doing only a few exercises lead a poor performance both on homeworks and on the exam exercises. This is called *underfitting*: you have a bad performance on the exam because you did not trained enough.

Moreover, brutally memorize all the homework lead to a perfect score on homeworks (trivially) but probably a bad score on the exam exercises. This is called *overfitting*: you have a bad performance on the exam because you did not captured the true essence of your homework, you have also memorize their "noise" (homework peculiarities).

Underfitting and overfitting - another example

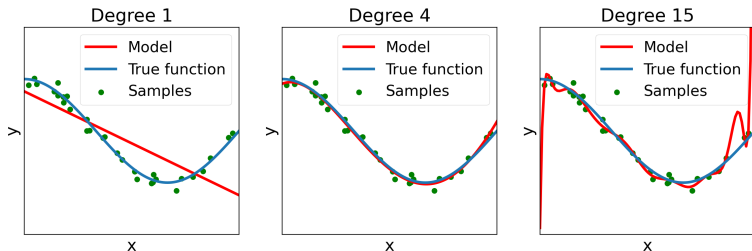


Figure: Underfitting (degree 1), good fitting (degree 4), overfitting (degree 15).

How to counter overfitting