Lecture 3: Linear regression

Introduction to machine learning

Kevin Webster

Department of Mathematics
Imperial College London

Outline

Outline

ML concepts review

In the previous lectures we introduced several key concepts in machine learning:

- Tasks, experience and performance measures
- Supervised and unsupervised learning
- Model capacity and regularisation
- Overfitting and underfitting
- Generalisation
- Model parameters and hyperparameters
- Model validation and selection

In addition, we have seen the k-NN algorithm as an example of supervised learning, and PCA as an example of unsupervised learning

Outline

Regression task

We will look more closely at many of these core concepts using the motivating example of linear regression.
Regression is a supervised learning task, where we assume we are given a dataset consisting of N input points

$$\mathbf{x} = (x_1, x_2, \ldots, x_N), \qquad x_i \in \mathbb{R}^d \quad \forall i$$

and N corresponding output values

$$\mathbf{y} = (y_1, y_2, \ldots, y_N), \qquad y_i \in \mathbb{R} \quad \forall i$$

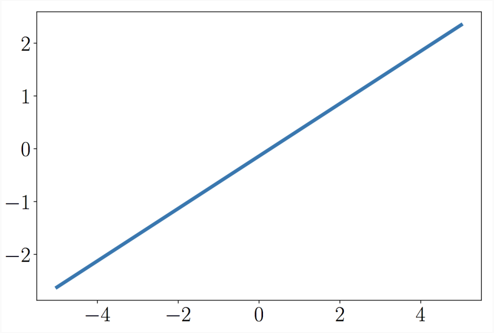
We wish to find a function f that models the relationship between $\mathbf{x}$ and $\mathbf{y}$. Typically f is a parametric function, depending on parameters $\boldsymbol{\theta}$.

Linear regression: setting

In the case of **linear regression**, we consider functions f that are linear in its parameters $\boldsymbol{\theta}$. For example, if $x \in \mathbb{R}$ then $f : \mathbb{R} \times \mathbb{R}^2 \mapsto \mathbb{R}$ given by

$$f(x, \boldsymbol{\theta}) = \theta_0 + \theta_1 x$$

is a linear regressor.

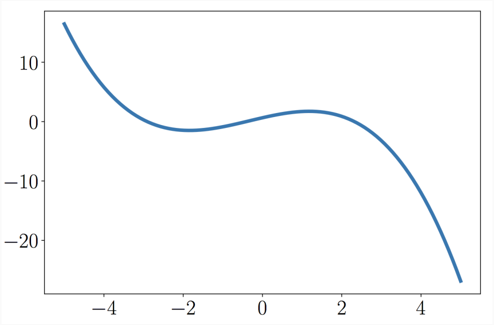


Linear regression: setting

However, note that we are requiring f to be linear in $\boldsymbol{\theta}$, not in x.

Therefore we also consider degree P polynomial regressors of the form $f : \mathbb{R} \times \mathbb{R}^{P+1} \mapsto \mathbb{R}$

$$f(x, \boldsymbol{\theta}) = \sum_{m=0}^{P} \theta_m x^m$$



Basis functions

The general form for our linear regressor is given by

$$f(x, \boldsymbol{\theta}) = \sum_{m=1}^{M} \theta_m \phi_m(x),$$

where the ϕ_i are called the **basis functions**. Note that the basis functions can be nonlinear in general.

- For linear functions (in x), the basis functions are given by $\{1, x\}$
- For univariate polynomial functions of degree P, the basis functions are $\{1, x, \ldots, x^P\}$
- Other basis functions are possible, such as radial basis functions (RBF)

Vector representation

Note that in each case above, we can represent the function f as an inner product between the parameter vectors and the feature vectors:

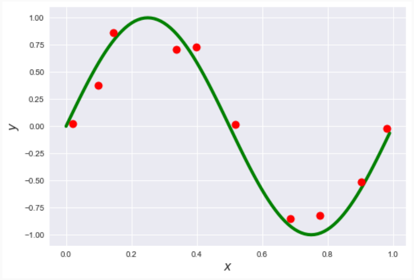
$$\begin{aligned} f(x,\boldsymbol{\theta}) &= \theta_0 + \theta_1 x = \begin{bmatrix} 1 & x \end{bmatrix} \begin{bmatrix} \theta_0 \\ \theta_1 \end{bmatrix} \\ f(x,\boldsymbol{\theta}) &= \sum_{m=0}^{P} \theta_m x^m = \underbrace{\begin{bmatrix} 1 & x & \cdots & x^P \end{bmatrix}}_{\text{feature vector}} \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_M \end{bmatrix} \end{aligned}$$

In both cases we have added the constant 1 to the feature vector. In the general case, the feature vector is given by

$$[\phi_1(x), \phi_2(x), \ldots, \phi_M(x)]$$

Noisy observations

- For any real dataset, we would not expect a linear (or even polynomial) regressor to fit the data exactly
- Our assumption is that the data is generated by an underlying regression model, but the observations are contaminated with noise
- As an example, the data below was generated by uniformly sampling x_n in the interval $[0, 1]$, computing the function $\sin(2\pi x)$ and adding a small amount of Gaussian-distributed noise



Noisy observations

- This ‘noisy data’ property is a modelling assumption
- It captures properties of many real datasets:
    - They contain an underlying regularity
    - Individual observations are corrupted by random noise
- The noise might come from part of the data generating process itself, i.e. some inherent stochasticity
- Alternatively, it may be due to measurement errors
- Most likely it is due to complex factors of variability that we are not able to capture in the data
- In our example above, our aim is to uncover the underlying regularity given by the function $\sin(2\pi x)$

Linear regression: setting

- It is common to assume the data is contaminated by noise
- Therefore, our linear regression model is given by

$$y = f(x, \boldsymbol{\theta}) + \epsilon$$

  where $x \in \mathbb{R}^d$, $\boldsymbol{\theta} \in \mathbb{R}^p$, and ϵ is a zero mean random variable with some predefined distribution that is independent of x
- Given a training set $(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)$, we seek optimal parameters $\boldsymbol{\theta}^*$
- Optimal in what sense?

Error function

- We wish to fit the function $f(x, \boldsymbol{\theta})$ to our available data
- We need to introduce a measure of how well the function fits the data
- This can be done by introducing an **error function** (or **loss function** that depends on the parameters $\boldsymbol{\theta}$
- The goal is to minimise this error function over the training data
- A widely-used choice of error function is the mean squared error (MSE):

$$L(\boldsymbol{\theta}) = \frac{1}{N} \sum_{n=1}^{N} \left(f(x_n, \boldsymbol{\theta}) - y_n\right)^2$$

- We will see later that this is a natural choice of error function for the given problem setting

MSE error function

Note that the error function

$$L(\boldsymbol{\theta}) = \frac{1}{N} \sum_{n=1}^{N} (f(x_n, \boldsymbol{\theta}) - y_n)^2$$

is a nonnegative quantity that would be zero if and only if the function $f(x, \boldsymbol{\theta})$ interpolated the data points.

Note also that (using the general form for $f(x, \boldsymbol{\theta})$),

$$L(\boldsymbol{\theta}) = \frac{1}{N} \sum_{n=1}^{N} \left(\sum_{m=1}^{M} \theta_m \phi_m(x_n) - y_n \right)^2$$

Design matrix

We define $\Phi_{\mathbf{x}} \in \mathbb{R}^{N \times M}$ by $\Phi_{\mathbf{x}}[i, j] = \phi_j(x_i)$. The matrix $\Phi_{\mathbf{x}}$ is called the **design matrix**. We also write $\mathbf{y} = [y_1, \ldots, y_N]^T \in \mathbb{R}^N$.

The design matrix consists of the feature vectors for every data point in the rows of the matrix.

Note that we can rewrite our error function more compactly as follows:

$$\begin{aligned} L(\boldsymbol{\theta}) &= \frac{1}{N} \sum_{n=1}^{N} \left(\sum_{m=1}^{M} \theta_m \phi_m(x_n) - y_n \right)^2 \\ &= \frac{1}{N} \left|\left|\Phi_{\mathbf{x}} \boldsymbol{\theta} - \mathbf{y}\right|\right|^2 \end{aligned}$$

Design matrix: example

For example, the design matrix for the following data

x	0.6	-1.1	0.3
y	3.6	0.3	0.9

for a regression function of the form $f(x, \boldsymbol{\theta}) = \theta_0 + \theta_1 x$ is given by

$$\Phi_{\mathbf{x}} = \begin{bmatrix} 1 & 0.6 \\ 1 & -1.1 \\ 1 & 0.3 \end{bmatrix}$$

Design matrix: example

A second example is given by the data

x_1	1.0	3.5	2.0	-0.2	0.3
x_2	0.5	0.1	0.8	1.1	0.7
y	-0.4	-12	-6.0	0.3	-1.0

with the second order polynomial regression function

$$f(x_1, x_2, \boldsymbol{\theta}) = \theta_0 x_1 + \theta_1 x_2 + \theta_2 x_1^2 + \theta_3 x_1 x_2 + \theta_4 x_2^2$$

and the design matrix

$$\Phi_{\mathbf{x}} = \begin{bmatrix} 1.0 & 0.5 & 1.0 & 0.5 & 0.25 \\ 3.5 & 0.1 & 12.25 & 0.35 & 0.01 \\ 2.0 & 0.8 & 4.0 & 1.6 & 0.64 \\ -0.2 & 1.1 & 0.04 & -0.22 & 1.21 \\ 0.3 & 0.7 & 0.09 & 0.21 & 0.49 \end{bmatrix}$$

Error function Hessian

Following the definition of our MSE error function, we have

$$\frac{\partial L}{\partial \theta_i} = \frac{2}{N} \sum_{n=1}^{N} \left(\sum_{m=1}^{M} \theta_m \phi_m(x_n) - y_n \right) \phi_i(x_n)$$

and

$$\frac{\partial^2 L}{\partial \theta_i \partial \theta_j} = \frac{2}{N} \sum_{n=1}^{N} \phi_i(x_n) \phi_j(x_n)$$

Therefore the Hessian is of the form

$$\frac{2}{N} \sum_{n=1}^{N} \boldsymbol{\phi}(x_n) \boldsymbol{\phi}(x_n)^T$$

where $\boldsymbol{\phi}(x) = [\phi_1(x), \phi_2(x), \ldots, \phi_M(x)]^T$.

Convex error function

Note that each term in the summand is a positive semi-definite matrix:

$$\boldsymbol{\eta}^T \phi(x_n)\phi(x_n)^T \boldsymbol{\eta} = \langle \boldsymbol{\eta}, \phi(x_n) \rangle^2 \geq 0$$

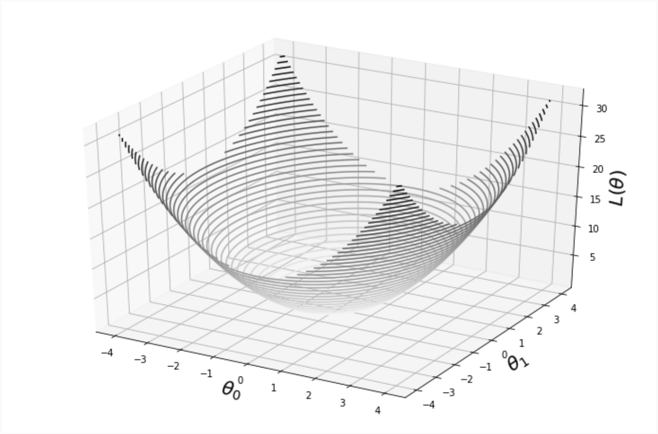
for all $\boldsymbol{\eta} \in \mathbb{R}^M$. So the Hessian is a sum of positive semi-definite matrices and thus is itself positive semi-definite. In fact, we can see that if the design matrix $\Phi_{\mathbf{x}}$ has full column rank, then the Hessian is positive definite (we return to this point later).

$\implies$ the error function is convex!

This means that the error function has a unique global minimum.

Convex error function

For the linear regressor $f(x, \boldsymbol{\theta}) = \theta_0 + \theta_1 x$, the error function can be visualised as follows:



Solution to least squares problem

We now derive the solution to our least squares problem.

Recall we have

$$\frac{\partial L}{\partial \theta_i} = \frac{2}{N} \sum_{n=1}^{N} \left(\sum_{m=1}^{M} \theta_m \phi_m(x_n) - y_n \right) \phi_i(x_n).$$

Setting the gradient to zero $\frac{\partial L}{\partial \boldsymbol{\theta}} = 0$ gives

$$\sum_{n=1}^{N} \phi_i(x_n) \left(\sum_{m=1}^{M} \theta_m \phi_m(x_n) \right) = \sum_{n=1}^{N} y_n \phi_i(x_n)$$

for all $i = 1, 2, \ldots, M$.

Solution to least squares problem

We have:

$$\sum_{n=1}^{N} \phi_i(x_n) \left(\sum_{m=1}^{M} \theta_m \phi_m(x_n) \right) = \sum_{n=1}^{N} y_n \phi_i(x_n), \qquad \forall i = 1, 2, \ldots, M$$

Recalling the notation $\mathbf{y} = [y_1, \ldots, y_N] \in \mathbb{R}^N$ and the design matrix $\Phi_{\mathbf{x}} \in \mathbb{R}^{N \times M}$ (defined by $\Phi_{\mathbf{x}}[i,j] = \phi_j(x_i)$), the above equation can be written in the more compact form:

$$\Phi_{\mathbf{x}}^T \Phi_{\mathbf{x}} \boldsymbol{\theta} = \Phi_{\mathbf{x}}^T \mathbf{y}$$

where $\Phi_{\mathbf{x}}^T \Phi_{\mathbf{x}} \in \mathbb{R}^{M \times M}$.

Solution to least squares problem

Now we have the condition for the global minimum of our optimisation problem:

$$\Phi_{\mathbf{x}}^T \Phi_{\mathbf{x}} \boldsymbol{\theta} = \Phi_{\mathbf{x}}^T \mathbf{y}$$

where $\Phi_{\mathbf{x}}^T \Phi_{\mathbf{x}} \in \mathbb{R}^{M \times M}$.

Therefore if $\Phi_{\mathbf{x}}^T \Phi_{\mathbf{x}}$ is invertible, the optimal parameters are given by

$$\boxed{\boldsymbol{\theta}^* = (\Phi_{\mathbf{x}}^T \Phi_{\mathbf{x}})^{-1} \Phi_{\mathbf{x}}^T \mathbf{y}}$$

This equation is called the **normal equation**. It gives the solution to the linear regression problem.

Under what conditions is the matrix $\Phi_{\mathbf{x}}^T \Phi_{\mathbf{x}}$ invertible?

Invertibility of $\Phi_{\mathbf{x}}^T \Phi_{\mathbf{x}}$

It is easy to see that $\Phi_{\mathbf{x}}^T \Phi_{\mathbf{x}}$ is a **Gram matrix**:

$$\Phi_{\mathbf{x}} = \begin{bmatrix} \langle v_1, v_1 \rangle & \langle v_1, v_2 \rangle & \cdots & \langle v_1, v_M \rangle \\ \langle v_2, v_1 \rangle & \langle v_2, v_2 \rangle & \cdots & \langle v_2, v_M \rangle \\ \vdots & \vdots & \ddots & \vdots \\ \langle v_M, v_1 \rangle & \langle v_M, v_2 \rangle & \cdots & \langle v_M, v_M \rangle \end{bmatrix}$$

where

$$v_i = [\phi_i(x_1), \phi_i(x_2), \ldots, \phi_i(x_N)]^T, \qquad i = 1, \ldots, M$$

A necessary and sufficient condition for a Gram matrix to be invertible is for the vectors v_i to be linearly independent.

Note this is precisely the condition that the design matrix has full column rank (recall earlier discussion on convexity of the loss function).

Invertibility of $\Phi_{\mathbf{x}}^T \Phi_{\mathbf{x}}$

$$v_i = [\phi_i(x_1), \phi_i(x_2), \dots, \phi_i(x_N)]^T, \qquad i = 1, \dots, M$$

Note the vector above captures one of the features for the whole dataset, for each i. These vectors will not be independent if:

- $M > N$. In this case the problem is overdetermined; for example this case arises if we try to fit a 10th degree polynomial to 5 data points
- There is redundancy in the features. For instance, if one feature is a linear combination of other features, this induces feature redundancy and the vectors v_i will be linearly dependent

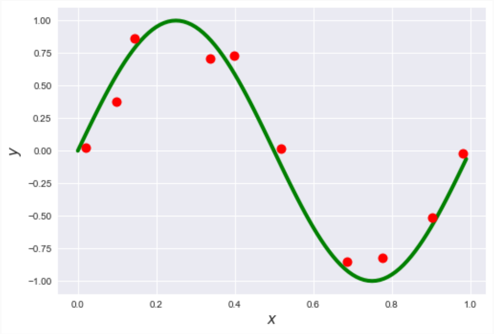
In either case above, the situation can be treated by removing redundant features. Note we are discounting the pathological case where all features are zero. From now on we assume that $\Phi_{\mathbf{x}}^T \Phi_{\mathbf{x}}$ is invertible.

Capacity

- The above discussion of the invertibility of $\Phi_{\mathbf{x}}^T \Phi_{\mathbf{x}}$ touches on the capacity of the linear regression model
- Note that the model capacity increases as we add more (linearly independent) features
- We need to choose the model capacity correctly according to the given dataset
- An appropriate capacity will depend on the size and complexity of the data
- The capacity (e.g. polynomial degree) is a hyperparameter of the model
- Low capacity gives strong **bias** and could lead to **underfitting**
- High capacity gives high **variance** and could lead to **overfitting**

Underfitting vs overfitting

We return to our previous example of data points generated by noisy observations of the function $\sin(2\pi x)$.

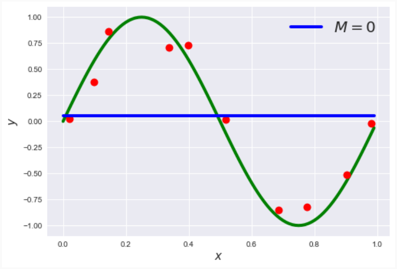


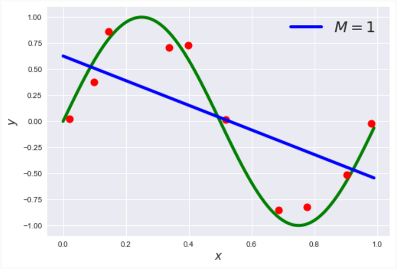
We consider using polynomial (of degree M) linear regressors of the form:

$$f(x, \boldsymbol{\theta}) = \sum_{m=0}^{M} \theta_m x^m$$

Underfitting vs overfitting

With $M = 0$ and $M = 1$ the regressor models are very simple (constant value and straight line respectively), and don't capture the data well:

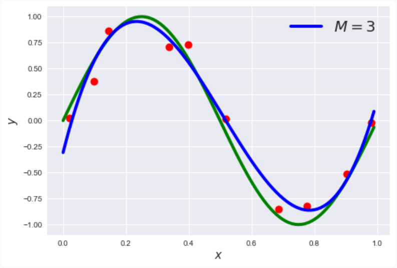


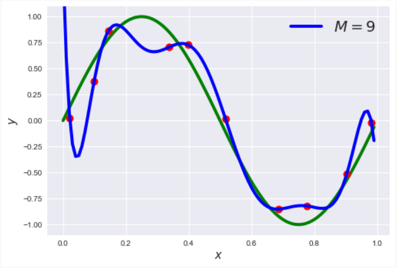


These models have low capacity, high bias and underfit the data.

Underfitting vs overfitting

Setting $M = 3$ gives a good fit to the data, but $M = 9$ overfits:

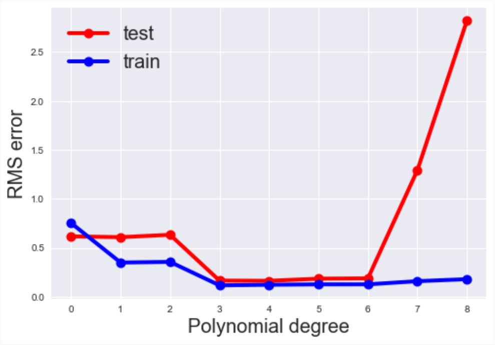




$M = 3$ has an appropriate capacity for this dataset, whereas the high capacity (and high variance) for $M = 9$ leads to overfitting.

Model validation and selection

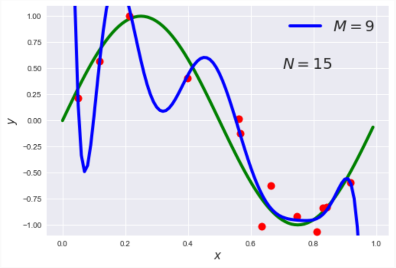
The polynomial degree M is a hyperparameter for these linear regression models. We can tune this hyperparameter using cross validation techniques.

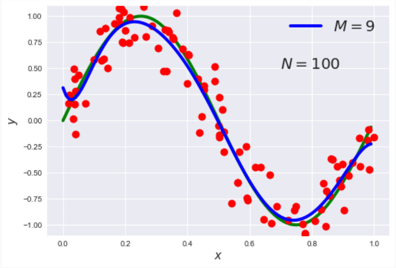


This following plot shows a typical behaviour of model performance according to different hyperparameter choices.

Underfitting vs overfitting

Remember that underfitting/overfitting depends on the data complexity and amount of data.





The $M = 9$ polynomial improves its approximation with more data. If the model capacity is sufficient, then more data reduces the overfitting problem.

Regularisation

- It is clearly far from ideal to choose the model hyperparameters according to the size of the available dataset
- Model overfitting can also be tackled using regularisation
- Regularisation is often applied by adding a penalty term to the error function:

$$L(\boldsymbol{\theta}) = \frac{1}{N}\sum_{n=1}^{N}\left(\sum_{m=1}^{M}\theta_m\phi_m(x_n) - y_n\right)^2 + \frac{\lambda}{N}\sum_{m=1}^{M}\theta_m^2$$

- This regularisation term discourages the coefficients θ_i from growing too large
- $\lambda \geq 0$ is the regularisation coefficient and is a hyperparameter. It controls the degree of regularisation

Regularisation

- The regularised loss expression can also be written

$$L(\boldsymbol{\theta}) = \frac{1}{N}\left(||\Phi_{\mathbf{x}}\boldsymbol{\theta} - \mathbf{y}||^2 + \lambda||\boldsymbol{\theta}||^2\right)$$

- This loss frequently appears in slightly different forms—the factor $\frac{1}{N}$ can also be omitted, or λ could be rescaled $\tilde{\lambda} = \lambda/N$
- When a quadratic penalty regularisation term is used (as above), the problem is often referred to as **ridge regression**
- In neural networks, this regulariser is often called **weight decay**

Solution to regularised least squares problem

The regularised least squares problem can also be solved in closed form. Similar to before, we compute

$$\frac{\partial L}{\partial \theta_i} = \frac{2}{N} \sum_{n=1}^{N} \left(\sum_{m=1}^{M} \theta_m \phi_m(x_n) - y_n \right) \phi_i(x_n) + \frac{2\lambda}{N} \theta_i$$

Setting the gradient to zero $\frac{\partial L}{\partial \boldsymbol{\theta}} = 0$ gives

$$\sum_{n=1}^{N} \phi_i(x_n) \left(\sum_{m=1}^{M} \theta_m \phi_m(x_n) \right) + \lambda \theta_i = \sum_{n=1}^{N} y_n \phi_i(x_n)$$

for all $i = 1, 2, \ldots, M$. This system of equations can be written

$$(\Phi_{\mathbf{x}}^T \Phi_{\mathbf{x}} + \lambda \mathbf{I}_M)\boldsymbol{\theta} = \Phi_{\mathbf{x}}^T \mathbf{y}$$

Solution to regularised least squares problem

As before, we can write the solution to the regularised least squares problem as

$$\boxed{\boldsymbol{\theta}^* = (\Phi_{\mathbf{x}}^T \Phi_{\mathbf{x}} + \lambda \mathbf{I}_M)^{-1} \Phi_{\mathbf{x}}^T \mathbf{y}}$$

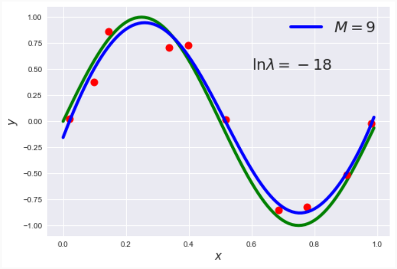
However, note this time that the matrix $(\Phi_{\mathbf{x}}^T \Phi_{\mathbf{x}} + \lambda \mathbf{I}_M)$ is always invertible for all $\lambda > 0$!

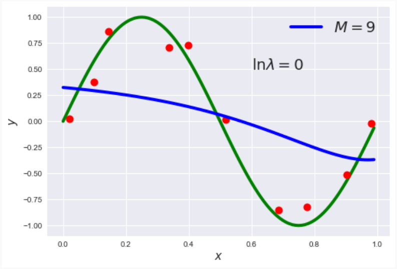
That is because $\Phi_{\mathbf{x}}^T \Phi_{\mathbf{x}}$ is a gram matrix and is therefore positive semi-definite, and adding $\lambda \mathbf{I}_M$ adds $\lambda > 0$ to all of its eigenvalues.

The regularised least squares problem always has a unique solution; this can also be seen by studying the Hessian of the regularised loss function.

Effect of regularisation

In our running example, the following plots show the effect of the regularisation parameter on the $M = 9$ polynomial function (that previously overfitted with no regularisation):





As before, the hyperparameter λ can be tuned with validation techniques.

Outline

Probabilistic setting

We return again to the setting of linear regression, this time with a probabilistic perspective.

We will show how the (regularised and unregularised) objective function can be derived in a principled way using a probabilistic approach.

Recall that we are given a dataset consisting of N input points

$$\mathbf{x} = (x_1, x_2, \ldots, x_N), \qquad x_i \in \mathbb{R}^d \quad \forall i$$

and N corresponding output values

$$\mathbf{y} = (y_1, y_2, \ldots, y_N), \qquad y_i \in \mathbb{R} \quad \forall i$$

We wish to find a parametric function $f(x, \boldsymbol{\theta})$ that models the relationship between $\mathbf{x}$ and $\mathbf{y}$.

Probabilistic setting

- We assume that the data is generated according to the following model
$$y = f(x, \boldsymbol{\theta}) + \epsilon, \quad \epsilon \sim \mathcal{N}(0, \sigma^2),$$
where σ^2 is a hyperparameter
- Therefore our assumption is that the data is generated by an underlying regression model, but observations are contaminated by Gaussian noise
- The random variable ϵ captures uncertainty due to inherent stochasticity of the process, measurement errors or just our ignorance of factors we are unable to capture

Likelihood function

A common approach in statistical modelling is to find the parameters that maximum the likelihood of the data.

The **likelihood function** is a function of the model parameters. It is defined as the probability of the data under the model, for a given set of parameters $\boldsymbol{\theta}$:

$$\mathcal{L}(\boldsymbol{\theta}|x, y) := p(y|x, \boldsymbol{\theta})$$

Here, $x \in \mathbb{R}^d$ is a data input and $y \in \mathbb{R}$ the corresponding output. $p(y|x, \boldsymbol{\theta})$ is the probability the model assigns to the input-output pair (x, y), which depends on the parameters $\boldsymbol{\theta}$.

Likelihood function

For our linear regression model, this likelihood is given by

$$\begin{aligned}
\mathcal{L}(\boldsymbol{\theta}|x,y) &= p(y|x,\boldsymbol{\theta}) \\
&= \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left[-\frac{(y-f(x,\boldsymbol{\theta}))^2}{2\sigma^2}\right] \\
&= \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left[-\frac{(y-\sum_{m=1}^{M}\theta_m\phi_m(x))^2}{2\sigma^2}\right]
\end{aligned}$$

since we have assumed that

$$y = f(x,\boldsymbol{\theta}) + \epsilon, \quad \epsilon \sim \mathcal{N}(0,\sigma^2),$$

Independent and identically distributed data

Our problem therefore translates into finding the optimal parameter values that maximise the likelihood function over the whole dataset:

$$\boldsymbol{\theta}_{ML} = \arg\max_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}|\mathbf{x}, \mathbf{y})$$

This is referred to as **maximum likelihood estimation (MLE)**.

A common assumption is that the data is **independent and identically distributed (i.i.d.)**, in which case we have

$$\begin{aligned} \mathcal{L}(\boldsymbol{\theta}|\mathbf{x}, \mathbf{y}) &= \prod_{i=1}^{N} p(y_i|x_i, \boldsymbol{\theta}) \\ &= \frac{1}{(2\pi\sigma^2)^{\frac{N}{2}}} \exp\left[-\sum_{i=1}^{N} \frac{(y_i - \sum_{m=1}^{M} \theta_m \phi_m(x_i))^2}{2\sigma^2} \right] \end{aligned}$$

Log-likelihood function

It is common to instead work with the **log-likelihood function** $\log \mathcal{L}(\boldsymbol{\theta}|x, y)$.

Since the logarithm is a monotonically increasing function of its argument, maximisation of the log-likelihood is equivalent to maximisation of the likelihood.

Taking the log also helps numerically, as multiplying small probabilities in the likelihood function could lead to numerical errors.
For our setting, we now have

$$\log \mathcal{L}(\boldsymbol{\theta}|\mathbf{x}, \mathbf{y}) = -\frac{1}{2\sigma^2} \sum_{i=1}^{N} (y_i - \sum_{m=1}^{M} \theta_m \phi_m(x_i))^2 - \frac{N}{2} \log(2\pi\sigma^2)$$

Maximum likelihood estimator

We have the log-likelihood function:

$$\log \mathcal{L}(\boldsymbol{\theta}|\mathbf{x}, \mathbf{y}) = -\frac{1}{2\sigma^2} \sum_{i=1}^{N} (y_i - \sum_{m=1}^{M} \theta_m \phi_m(x_i))^2 - N \log \sqrt{2\pi} - N \log \sigma$$

Our task is to find the maximum likelihood estimator $\boldsymbol{\theta}^*$ that maximises the above. Equivalently, we can consider to minimise the negative log-likelihood (NLL). We can see that

$$\boldsymbol{\theta}_{ML} = \arg\min_{\boldsymbol{\theta}} [-\log \mathcal{L}(\boldsymbol{\theta}|\mathbf{x}, \mathbf{y})] = \arg\min_{\boldsymbol{\theta}} \frac{1}{N} \sum_{i=1}^{N} (y_i - \sum_{m=1}^{M} \theta_m \phi_m(x_i))^2$$

$\implies$ the solution to the least squares problem is precisely the MLE!

Maximum a posteriori (MAP) estimation

- The maximum likelihood estimator is a commonly used frequentist estimator
- We have already seen how this estimator is susceptible to overfitting
- We already introduced the concept of regularisation to tackle overfitting
- We can also interpret regularisation from a probabilistic (Bayesian) perspective
- We start by introducing a **prior distribution** on the coefficients $\boldsymbol{\theta}$:

  $$\boldsymbol{\theta} \sim \mathcal{N}(\boldsymbol{\theta}|\mathbf{0}, \alpha^2 \mathbf{I}_M)$$

  for some choice of hyperparameter α
- The choice of prior reflects our uncertainty and/or ignorance about the true parameters $\boldsymbol{\theta}$

Maximum a posteriori (MAP) estimation

- We are interested in finding the probable values of $\boldsymbol{\theta}$ given the data
- By Bayes rule, we know that

$$p(\boldsymbol{\theta}|\mathbf{x}, \mathbf{y}) = \frac{p(\mathbf{y}|\mathbf{x}, \boldsymbol{\theta})p(\boldsymbol{\theta})}{\int_{\boldsymbol{\theta}} p(\mathbf{y}|\mathbf{x}, \boldsymbol{\theta})d\boldsymbol{\theta}}$$

- This can be summarised by representing the data as $\mathcal{D}$:

$$p(\boldsymbol{\theta}|\mathcal{D}) = \frac{p(\mathcal{D}|\boldsymbol{\theta})p(\boldsymbol{\theta})}{p(\mathcal{D})}$$

- The distribution $p(\boldsymbol{\theta}|\mathcal{D})$ is the **posterior distribution**, and represents our updated belief about the parameters $\boldsymbol{\theta}$ given the data

Maximum a posteriori (MAP) estimation

We can determine the most likely value of $\boldsymbol{\theta}$ given the data by maximising $p(\boldsymbol{\theta}|\mathbf{x}, \mathbf{y})$.

Note that the denominator on the right hand side does not depend on $\boldsymbol{\theta}$, and so we have

$$p(\boldsymbol{\theta}|\mathbf{x}, \mathbf{y}) \propto p(\mathbf{y}|\mathbf{x}, \boldsymbol{\theta})p(\boldsymbol{\theta})$$

Therefore we have

$$\begin{aligned}\boldsymbol{\theta}_{MAP} &= \arg\max_{\boldsymbol{\theta}} p(\boldsymbol{\theta}|\mathbf{x}, \mathbf{y}) \\ &= \arg\max_{\boldsymbol{\theta}} p(\mathbf{y}|\mathbf{x}, \boldsymbol{\theta})p(\boldsymbol{\theta})\end{aligned}$$

Maximum a posteriori (MAP) estimation

We can again recast the optimisation problem by taking the negative logarithm:

$$\boldsymbol{\theta}_{MAP} = \arg\min_{\boldsymbol{\theta}} \left[-\log p(\mathbf{y}|\mathbf{x}, \boldsymbol{\theta}) - \log p(\boldsymbol{\theta}) \right]$$

Note that the first term above is the negative log-likelihood we saw earlier.

We have already shown how this term can be written as

$$-\log p(\mathbf{y}|\mathbf{x}, \boldsymbol{\theta}) = \frac{1}{2\sigma^2} \sum_{i=1}^{N} (y_i - \sum_{m=1}^{M} \theta_m \phi_m(x_i))^2 + N \log \sqrt{2\pi}$$

Maximum a posteriori (MAP) estimation

In a similar way, we can write

$$
\begin{aligned}
-\log p(\boldsymbol{\theta}) &= -\log \left\{ \frac{1}{(2\pi\alpha^2)^{\frac{M}{2}}} \exp\left[-\frac{\boldsymbol{\theta}^T\boldsymbol{\theta}}{2\alpha^2} \right] \right\} \\
&= \frac{1}{2\alpha^2}\boldsymbol{\theta}^T\boldsymbol{\theta} + \frac{M}{2}\log(2\pi\alpha^2)
\end{aligned}
$$

The above follows from the probability density function of the multivariate Gaussian distribution.

Note that as before, the second term does not depend on $\boldsymbol{\theta}$ and can therefore be omitted in the formulation of the optimisation objective.

Maximum a posteriori (MAP) estimation

Bringing both terms together, we find

$$\begin{aligned}
\boldsymbol{\theta}_{MAP} &= \arg\min_{\boldsymbol{\theta}} \left[-\log p(\mathbf{y}|\mathbf{x},\boldsymbol{\theta}) - \log p(\boldsymbol{\theta})\right] \\
&= \arg\min_{\boldsymbol{\theta}} \left\{ \frac{1}{2\sigma^2} \sum_{i=1}^{N} (y_i - \sum_{m=1}^{M} \theta_m \phi_m(x_i))^2 + \frac{1}{2\alpha^2} \boldsymbol{\theta}^T \boldsymbol{\theta} \right\} \\
&= \arg\min_{\boldsymbol{\theta}} \left\{ \frac{1}{N} \sum_{i=1}^{N} (y_i - \sum_{m=1}^{M} \theta_m \phi_m(x_i))^2 + \frac{\sigma^2}{N\alpha^2} \sum_{m=1}^{M} \theta_m^2 \right\}
\end{aligned}$$

and we see that the solution to the regularised least squares problem is precisely the MAP estimate of the parameters $\boldsymbol{\theta}$, with the regularisation parameter given by $\lambda = \sigma^2/\alpha^2$.

Outline

Linear algebra view of linear regression

In this section we will look at linear regression from a linear algebra perspective and again derive the equations for the problem solution.

Recall our problem setting; we have the data

$$\mathbf{x} = (x_1, x_2, \ldots, x_N), \quad \mathbf{y} = (y_1, y_2, \ldots, y_N)$$

where $x_i \in \mathbb{R}^d$, $y_i \in \mathbb{R}\ \forall i$. The general form of our linear regressor is

$$f(x, \boldsymbol{\theta}) = \sum_{m=1}^{M} \theta_m \phi_m(x) = [\phi_1(x), \phi_2(x), \ldots, \phi_M(x)] \begin{bmatrix} \theta_1 \\ \theta_2 \\ \vdots \\ \theta_M \end{bmatrix}$$

Linear algebra view of linear regression

Our model predictions for all input values $\mathbf{x}$ is given by

$$\Phi_{\mathbf{x}}\boldsymbol{\theta} \in \mathbb{R}^N,$$

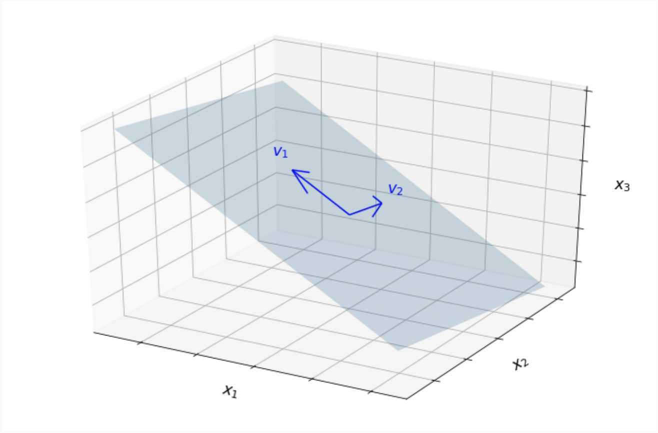
where $\Phi_{\mathbf{x}}$ is the design matrix and $\boldsymbol{\theta} = [\theta_1, \ldots, \theta_M]^T \in \mathbb{R}^M$. We would like our model predictions to be as close as possible to the true output values $\mathbf{y} \in \mathbb{R}^N$.

In general we cannot expect $\mathbf{y}$ to lie in the column space of $\Phi_{\mathbf{x}}$ (the usual situation is $N \gg M$). Therefore we choose to seek $\boldsymbol{\theta}^*$ that minimises the distance

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} ||\Phi_{\mathbf{x}}\boldsymbol{\theta} - \mathbf{y}||^2_{\mathbb{R}^N}$$

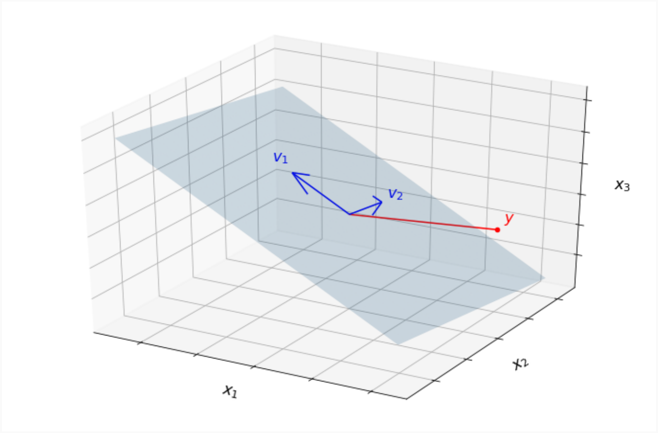
Linear algebra view of linear regression

As an example, suppose $M = 2$ and we have 3 data points. We write $\Phi_{\mathbf{x}} = [v_1|v_2]$ and the column space (or image of $\Phi_{\mathbf{x}}$) can be visualised as follows:



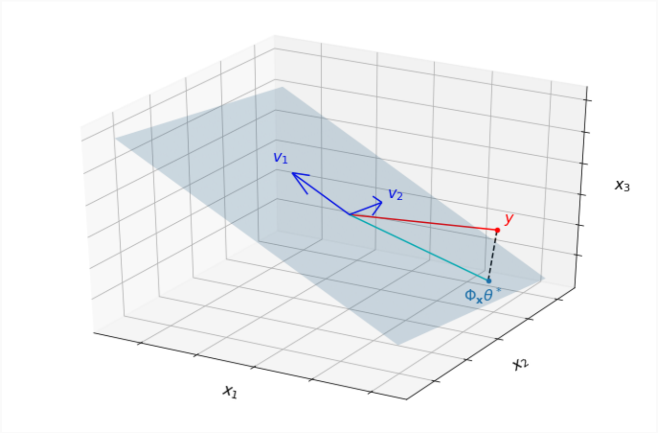
Linear algebra view of linear regression

In general, $\mathbf{y}$ will not lie in $\operatorname{Im}\Phi_{\mathbf{x}}$ (so there is no $\boldsymbol{\theta}$ such that $\Phi_{\mathbf{x}}\boldsymbol{\theta} = \mathbf{y}$):



Linear algebra view of linear regression

The linear regression problem seeks the parameters $\boldsymbol{\theta}^*$ that minimise $||\Phi_{\mathbf{x}}\boldsymbol{\theta} - \mathbf{y}||^2$, or equivalently the distance between model predictions and $\mathbf{y}$.



Linear algebra view of linear regression

We will derive the solution using only tools from linear algebra. First, we need the following Lemma:

Lemma
For any $\mathbf{A} \in \mathbb{R}^{m \times n}$, $\mathbf{x} \in \mathbb{R}^n$ *and* $\mathbf{y} \in \mathbb{R}^m$, *we have*

$$\langle \mathbf{A}\mathbf{x}, \mathbf{y} \rangle_{\mathbb{R}^m} = \langle \mathbf{x}, \mathbf{A}^T \mathbf{y} \rangle_{\mathbb{R}^n}$$

Proof.
Left as an exercise. $\square$

Linear algebra view of linear regression

An equivalent way of viewing the previous Lemma is to write

$$(\operatorname{Im} \mathbf{A})^{\perp} = \ker \mathbf{A}^{T}$$

since (assuming $\mathbf{y} \neq \mathbf{0}$ as clearly $\{\mathbf{0}\} \in (\operatorname{Im} \mathbf{A})^{\perp} \cap \ker \mathbf{A}^{T}$):

$$\begin{aligned}
\mathbf{y} \in \operatorname{Im}(\mathbf{A})^{\perp} &\Leftrightarrow \langle \mathbf{A}\mathbf{x}, \mathbf{y} \rangle = 0 \quad \forall \mathbf{x} \in \mathbb{R}^{n} \\
&\Leftrightarrow \langle \mathbf{x}, \mathbf{A}^{T}\mathbf{y} \rangle = 0 \quad \forall \mathbf{x} \in \mathbb{R}^{n} \\
&\Leftrightarrow \mathbf{A}^{T}\mathbf{y} = 0 \\
&\Leftrightarrow \mathbf{y} \in \ker(\mathbf{A}^{T})
\end{aligned}$$

Linear algebra view of linear regression

We assume that the design matrix $\Phi_{\mathbf{x}}$ has full column rank (see earlier discussion). Note then that the image of $\Phi_{\mathbf{x}}$ is an M-dimensional subspace in $\mathbb{R}^N$. We define the (invertible!) mapping

$$\Phi_{\mathbf{x}}\big|_{\operatorname{Im}\Phi_{\mathbf{x}}} : \mathbb{R}^M \mapsto \operatorname{Im}\Phi_{\mathbf{x}}$$

by simply restricting the codomain to the image of $\Phi_{\mathbf{x}}$.

We also define the orthogonal projection

$$\pi : \mathbb{R}^N \mapsto \operatorname{Im}\Phi_{\mathbf{x}}$$

Linear algebra view of linear regression

Now note that

$$\begin{aligned} ||\Phi_{\mathbf{x}}\boldsymbol{\theta} - \mathbf{y}||^2 &= ||\Phi_{\mathbf{x}}\boldsymbol{\theta} - \pi\mathbf{y}||^2 + 2\langle \underbrace{\Phi_{\mathbf{x}}\boldsymbol{\theta} - \pi\mathbf{y}}_{\in \operatorname{Im}\Phi_{\mathbf{x}}}, \underbrace{(\pi - I_N)\mathbf{y}}_{\in(\operatorname{Im}\Phi_{\mathbf{x}})^T} \rangle + ||(\pi - I_N)\mathbf{y}||^2 \\ &= ||\Phi_{\mathbf{x}}\boldsymbol{\theta} - \pi\mathbf{y}||^2 + ||(\pi - I_N)\mathbf{y}||^2 \end{aligned}$$

so the norm $||\Phi_{\mathbf{x}}\boldsymbol{\theta} - \mathbf{y}||^2$ is minimised when $\boldsymbol{\theta}$ is chosen such that $\Phi_{\mathbf{x}}\boldsymbol{\theta} = \pi\mathbf{y}$, since the second term is independent of $\boldsymbol{\theta}$.

This is possible since π is the projection onto $\operatorname{Im}\Phi_{\mathbf{x}}$.

Linear algebra view of linear regression

Let $\Phi_{\mathbf{x}}^{\dagger} = (\Phi_{\mathbf{x}}|_{\mathrm{Im}(\Phi_{\mathbf{x}})})^{-1}\pi : \mathbb{R}^N \to \mathbb{R}^2$. Then

$$\begin{aligned}
\boldsymbol{\theta} = \Phi_{\mathbf{x}}^{\dagger}\mathbf{y} \quad &\Leftrightarrow \quad \Phi_{\mathbf{x}}\boldsymbol{\theta} = \pi\mathbf{y} \\
&\Leftrightarrow \quad (\Phi_{\mathbf{x}}\boldsymbol{\theta} - \mathbf{y}) \in \mathrm{Im}(\Phi_{\mathbf{x}})^{\perp} \\
&\Leftrightarrow \quad (\Phi_{\mathbf{x}}\boldsymbol{\theta} - \mathbf{y}) \in \ker(\Phi_{\mathbf{x}}^{T}) \\
&\Leftrightarrow \quad \Phi_{\mathbf{x}}^{T}(\Phi_{\mathbf{x}}\boldsymbol{\theta} - \mathbf{y}) = \mathbf{0} \\
&\Leftrightarrow \quad \boldsymbol{\theta} = (\Phi_{\mathbf{x}}^{T}\Phi_{\mathbf{x}})^{-1}\Phi_{\mathbf{x}}^{T}\mathbf{y}
\end{aligned}$$

and we have recovered the normal equation!