NPCW
2025

Aalto University

Probabilistic machine learning | Intro

Introduction to machine learning

Francesco Corona

Chemical and Metallurgical Engineering
School of Chemical Engineering

We all know curve fitting

We consider a classic **regression** problem and use it to introduce some central concepts

- Suppose we observe the value $x \in \mathcal{X} \subseteq \mathcal{R}$ of some variable X that we call **input**
- We wish to use x to estimate the value $t \in \mathcal{T} \subseteq \mathcal{R}$ of a variable T, the **target**

We are implicitly assuming that there exists some function relating variables X and T

- However, such a relation between X and T is unknown to us
- We wish to recover it, in some sense, and use it for prediction

At our disposal, we only have a collection of N pairs, $\{(x_n, t_n)\}_{n=1}^{N}$, the **training dataset**

Input data	Target data
x_1	t_1
x_2	t_2
$\vdots$	$\vdots$
x_n	t_n
$\vdots$	$\vdots$
x_N	t_N

We assume

$$f : \mathcal{X} \to \mathcal{T} \qquad (f : x \mapsto t)$$

We want

$$\widehat{f} : \mathcal{X} \to \mathcal{T}$$

How?

January 15, 2025
FC

Curve fitting | Example

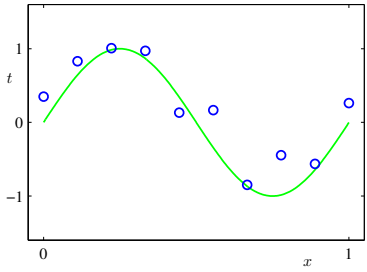
Let us develop the theory using data generated from a function consisting of two parts

- Some deterministic term
- An additive noise term

We shall assume that the training data consists of $N = 10$ pairs (x_n, t_n) generated by

$$t = \underbrace{\sin(2\pi x) + \text{noise}}_{f}$$

Each of the training examples (blue circles) is a point comprising an observation x_n of the input variable X along with the associated observation t_n of the target variable T



The deterministic function $\sin(2\pi x)$ (green curve) and a small amount of Gaussian noise is used to generate the data $\{(x_n, t_n)\}$

How to estimate the value of t for some x?

- ... without knowing the green curve and the nature of the noise term

Curve fitting | Example (cont.)

January 15, 2025
– FC –

This is an intrinsically hard problem to solve, as we have to generalise from a finite set

- Moreover, for a given x there is uncertainty as to the proper value for t

In probabilistic machine learning, we will build on **probability theory** to provide a modelling framework for expressing such uncertainty in a precise and quantitative manner

Then, we use **decision theory** to produce estimates which are optimal, in some sense

However, we are not interested in rushing up the theory and we shall proceed informally

Curve fitting | Example | Model

We consider a simple approach to curve fitting based on a certain polynomial function

$$y(x|w) = w_0 + w_1 x + w_2 x^2 + \cdots + w_{n_m} x^{n_m} + \cdots w_{N_m} x^{N_m}$$
$$= \sum_{n_m=0}^{N_m} w_{n_m} x^{n_m}$$

N_m is the order of the polynomial and x^{n_m} is the value of x raised to the power of n_m

The coefficients $\{w_{n_m}\}_{n_m=0}^{N_m}$ are collected in a parameter vector $w \in \mathcal{R}^{N_m}$

January 15, 2025

The function $y(x|w)$ is a modelling choice (how we choose to represent the function f)

- It is a nonlinear function of the input values x
- It is a linear function of the parameters w

Functions like $y(x|w)$ that are linear in the parameters are **linear models** for regression

- The parameters w which characterise $y(x|w)$ can be estimated from data $\{(x_n, y_n)\}$

Curve fitting | Example | Model | Accuracy

$$y(x|w) = \sum_{n_m=0}^{N_m} w_{n_m} x^{n_m}$$

The values of the parameters are estimated by fitting the polynomial model to the data

- We look for the set of parameters w that makes $y(x|w)$ best matches the data

This is done by minimising an **error** function, a function of the unknown parameters w that quantifies the mismatch between model outputs $y(x_n|w)$ and training outputs t_n

- ... cumulatively, over the entire set $\{(x_n, t_n)\}_{n=1}^{N}$ of training data

A common choice of error function is the **sum of the squares of the errors** on $\{(x_n, t_n)\}$

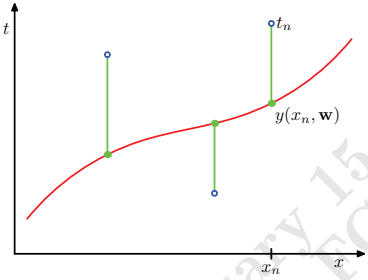
$$E(w) = \left(\frac{1}{2}\right) \sum_{n=1}^{N} \Big(y(x_n|w) - t_n\Big)^2$$

It is clear that $E(w)$ is a non-negative quantity that would be equal to zero if and only if the model function $y(x|w)$ were to pass through each of the training data $\{(x_n, t_n)\}$

January 15, 2025 - IIC -

Curve fitting | Example | Model | Accuracy (cont.)

Graphically, error function $E(w)$ corresponds to (one half of) the sum of the squares of the mismatches (vertical green bars) between each data point t_n and the model $y(x|w)$



$$E(w) = \frac{1}{2}\sum_{n=1}^{N}\Big(y(x_n|w) - t_n\Big)^2$$

To solve the curve fitting task, we search for the w for which $E(w)$ is the smallest

January 15, 2025
FC

Note that because the error function is a quadratic function of the parameters w, thence its derivatives with respect to the parameters are linear functions in the elements of w

- The minimisation of $E(w)$ has the unique analytic minimiser $w^\star$
- (We get the solution $w^\star$ by solving a set of linear equations)

The optimal (for the chosen error function) linear regression model is the polynomial

$$y(x|w^\star) = \sum_{n_m=0}^{N_m} w^\star_{n_m} x^{n_m}$$

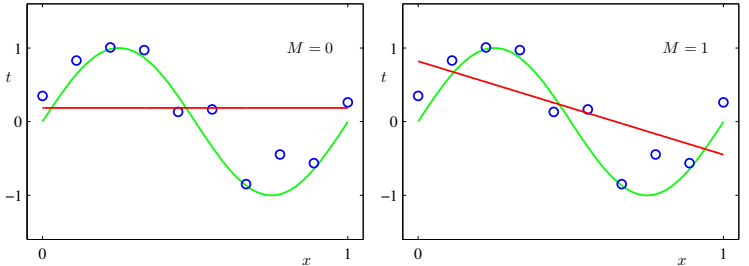
Curve fitting | Example | Model | Complexity

$$y(x|w) = w_0 + w_1 x + w_2 x^2 + \cdots + w_{n_m} x^{n_m} + \cdots w_{N_m} x^{N_m}$$
$$= \sum_{n_m=0}^{N_m} w_{n_m} x^{n_m}$$

For the user, there still remains the problem of choosing the order N_m of the polynomial

- This task is known as **model comparison** or **model selection**

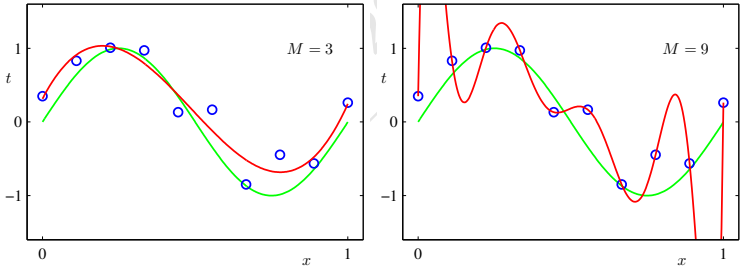
Different orders N_m of $y(x|\{w_{n_m}\}_{n_m=0}^{N_m})$ on data $\{(x_n, t_n)\}_{n=1}^{N}$ will give different results



Constant ($N_m = 0$) and first order ($N_m = 1$) polynomials yield a poor fit to the data

Curve fitting | Example | Model | Complexity (cont.)

A third order ($N_m = 3$) polynomial gives a good fit to data, whereas the higher order polynomial ($N_m = 9$) gives an excellent fit, although the fitted curve oscillates wildly



For $N_m = 9$, the polynomial passes exactly through each training point and $E(w^\star) = 0$

This behaviour, known as **over-fitting**, associates with a poor generalisation performance

We all know curve fitting (cont.)

January 15, 2025

The reasonable question to ask is how to make accurate predictions also for new data?

We can obtain some quantitative insight into the dependence of the generalisation performance on the polynomial order N_m by considering a separate, unseen, **test dataset**

For each choice of N_m, we evaluate the error $E(w^\star)$ for both training and test dataset

Curve fitting | Example | Model | Complexity (cont.)

$$E(w) = \frac{1}{2}\sum_{n=1}^{N}\Big(y(x_n|w) - t_n\Big)^2$$

It is sometimes convenient to utilise the **square root of the mean squared error**, or E_{RMS}

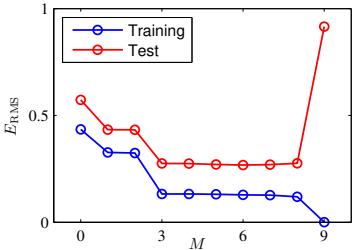
$$E_{RMS}(w^\star) = \sqrt{\frac{(2)E(w^\star)}{N}}$$

The division by N allows a consistent comparison of datasets of different sizes and the square root ensures that $E_{RMS}(w^\star)$ is on the same scale/units as the target variables

January 15, 2025

Curve fitting | Example | Model | Complexity (cont.)

The $E_{RMS}(w^\star)$ evaluated on the training and test datasets, for various values of N_m



- Small values of N_m (0, 1 and 2) yield relatively large errors on the test set
- $3 \leq N_m \leq 8$ yield small test errors

Low-order polynomials are rather inflexible and incapable of capturing the deterministic part of the underlying function, whereas high-order polynomials are too flexible and able of capturing also variations associated to the randomness added to that function

For $N_m = 9$, the training error is zero, because this polynomial has 10 degrees of freedom corresponding to $|\{w_{n_m}\}|$ and thus can be tuned exactly to the $N = 10$ data

- The test error has become very large

Curve fitting | Example | Model | Complexity (cont.)

January 10, 2025

We gain insight into the issue by analysing the estimates of the model parameters $w^\star$

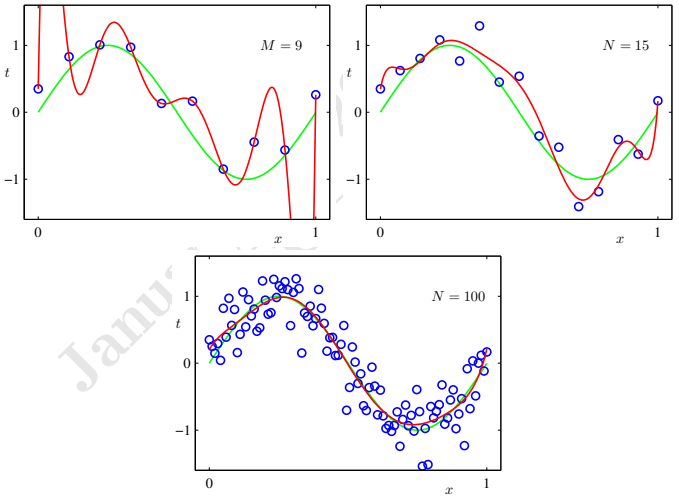
	$M=0$	$M=1$	$M=6$	$M=9$
$w_0^\star$	0.19	0.82	0.31	0.35
$w_1^\star$		-1.27	7.99	232.37
$w_2^\star$			-25.43	-5321.83
$w_3^\star$			17.37	48568.31
$w_4^\star$				-231639.30
$w_5^\star$				640042.26
$w_6^\star$				-1061800.52
$w_7^\star$				1042400.18
$w_8^\star$				-557682.99
$w_9^\star$				125201.43

As N_m increases, $|w^\star|$ typically gets larger

- Large positive/negative values

Curve fitting | Example | Model | Complexity (cont.)

We can also study the behaviour of a model of fixed complexity ($N_m = 9$) as N varies



Over-fitting is less severe with large N and larger N can afford us more complex models

Curve fitting | Example | Model | Complexity (cont.)

January 1st 2025
_ IIC _

The heuristic from heaven (hell?) recites 'the number of data points should be no less than some multiple (5, 10, or 100) of the number of tuning parameters in the model' Yet, the number of parameters is not necessarily the best measure of model complexity

- Why limit the number of parameters according to the size of the training set?
- Why not choose model complexity according to the problem complexity?

Curve fitting | Example | Model | Accuracy and complexity

We shall see that least squares minimisation for fitting model parameters is a specific case of **likelihood maximisation** and that over-fitting is understood as one of its properties

- Over-fitting can be avoided by adopting a full probabilistic (**Bayesian**) approach

We shall also see that there is no difficulty from a Bayesian perspective in employing models whose number of parameters greatly exceeds the size of the training data set

- The **effective number of parameters** adapts automatically to the data size

January 15, 2025
ITC

Curve fitting | Example | Model | Regularisation

Before getting there, we continue with our approach and consider how we can apply it to data sets of limited size, while still choosing to use some relatively complex model

One popular technique, known as **regularisation**, is used to gain control over over-fitting

- To discourage the model parameters from taking on large values, regularisation approaches involve adding a **penalty term** to the standard error function ($E(w)$)
- The simplest such penalty term takes the form of a sum of squares of all parameters

$$\widetilde{E}(w|\lambda) = \frac{1}{2}\sum_{n=1}^{N}\Big(y(x_n|w) - t_n\Big)^2 + \frac{\lambda}{2}|w|^2$$

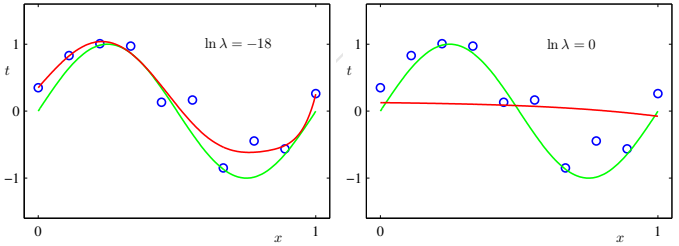
$|w|^2 = w^T w = w_0^2 + \cdots + w_{n_m}^2 + \cdots + w_{N_m}^2$ and the (**hyper-**) **parameter** λ governs the relative importance of the regularisation term, compared with the sum-of-squares term

Notably, such a regularised error function $\widetilde{E}(w|\lambda)$ can also be minimised analytically

Curve fitting | Example | Model | Regularisation (cont.)

$$\widetilde{E}(w|\lambda) = \frac{1}{2}\sum_{n=1}^{N}\Big(y(x_n|w) - t_n\Big)^2 + \frac{\lambda}{2}|w|^2$$

Fitting the polynomial of order $N_m = 9$ to the training data using a regularised error



- With $\ln\lambda = -18$ (a small value of λ), over-fitting is suppressed
- With $\ln\lambda = 0$ (a large value of λ), we obtain again a poor fit

$$y(x|w) = w_0 + w_1 x + w_2 x^2 + \cdots + w_{N_m} x^{N_m=9}$$

Curve fitting | Example | Model | Regularisation (cont.)

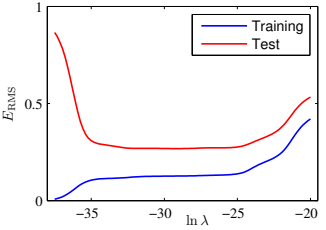
The parameters $w^\star$ for order $N_m = 9$ polynomials with varying regularisation term λ

	$\ln\lambda = -\infty$	$\ln\lambda = -18$	$\ln\lambda = 0$
$w_0^\star$	0.35	0.35	0.13
$w_1^\star$	232.37	4.74	-0.05
$w_2^\star$	-5321.83	-0.77	-0.06
$w_3^\star$	48568.31	-31.97	-0.05
$w_4^\star$	-231639.30	-3.89	-0.03
$w_5^\star$	640042.26	55.28	-0.02
$w_6^\star$	-1061800.52	41.32	-0.01
$w_7^\star$	1042400.18	-45.95	-0.00
$w_8^\star$	-557682.99	-91.53	0.00
$w_9^\star$	125201.43	72.68	0.01

- Un-regularised model, for $\ln\lambda = -\infty$
- As λ increases, $|w^\star|^2$ gets smaller

Regularisation reduces the magnitude of the parameters at the expenses of accuracy

Impact of penalty term λ on the regularised root-mean-square error $\widetilde{E}_{RMS}(w^\star|\lambda)$



$$\widetilde{E}_{RMS}(w^\star|\lambda) = \sqrt{\frac{2\widetilde{E}(w^\star|\lambda)}{N}}$$

May 26, 2025
PC_

Curve fitting | Example | Model | Cross-validation

January 15, 2025
FC

Before minimising an error function, the user must determine a suitable degree of model complexity, whether this is a polynomial order N_m or the regularisation parameter λ

A way of achieving this is to take all the available data and partition it into two subsets

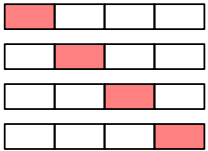
- A training set to determine $w^\star$ for various settings of model complexity, N_m or λ
- A **validation set** to determine a suitable value of model complexity, $N_m^\star$ or $\lambda^\star$

Seeing the inherent data scarcity, this may seem too wasteful of valuable training data

- A common workaround is to use a technique known as **cross-validation**

Curve fitting | Example | Model | Cross-validation (cont.)

S-**fold cross-validation** allows a proportion $(S-1)/S$ of the available data to be used for training while making use of all of the reminder of the data to asses performance



For the $S = 4$ case, we partition the data into S groups

- In the simplest case blocks are of equal size

$S-1$ folds (the white blocks) are used to train a set of models and the held-out fold (the red block) is for testing

This procedure is repeated independently for all S possible choices of the held-out fold

- The performance scores from all the S runs are then averaged

When data are particularly scarce, it is appropriate to consider the extreme case $S = N$

- This procedure is known as **leave-one-out** (**LOO**) **cross-validation**

January 15, 2025 – FC –