Data approximation and quadrature

Francesco Marchetti

Fundamentals of Computational Mathematics

Summary

Least-squares approximation: introduction

We introduce least-squares approximation from the most basic setting.
Suppose that m observations $y_1, \dots, y_m$ of a certain process at times $t_1, \dots, t_m$ are at disposal (e.g. the value of a stock at different times). Our aim is to find the straight line $y = c_1 + c_2 x$ that *better* fit the data, being c_1, c_2 parameters.
In principle, if the line intersects each couple (t_i, y_i), $i = 1, \dots, m$, we find the best model. Translating this into a linear system becomes

$$\begin{cases} c_1 + c_2 t_1 = y_1 \\ \vdots \\ c_1 + c_2 t_m = y_m. \end{cases}$$

Least-squares approximation: introduction (cont.)

We can rewrite the linear system as $A\boldsymbol{c} = \boldsymbol{y}$, where

$$A = \begin{pmatrix} 1 & t_1 \\ \vdots & \vdots \\ 1 & t_m \end{pmatrix}, \ \boldsymbol{c} = (c_1, c_2)^\mathsf{T}, \ \boldsymbol{y} = (y_1, \ldots, y_m)^\mathsf{T}.$$

This linear system is overdetermined, and almost surely has no solution (three non-collinear points are sufficient...).
Therefore, we need to establish what we mean with a *good fit*.
Here, we consider the **least-squares** approach: letting $\boldsymbol{r} = \boldsymbol{r}(\boldsymbol{c}) = A\boldsymbol{c} - \boldsymbol{y}$, we consider the minimization problem

$$\min_{\boldsymbol{c} \in \mathbb{R}^2} \frac{1}{2} \boldsymbol{r}^\mathsf{T} \boldsymbol{r} = \frac{1}{2} \sum_{i=1}^{m} r_i^2.$$

Doing in this way, we find the line that minimize the sum of square residuals at times t_i, $i = 1, \ldots, m$.

Least-squares approximation: generalization

In a more general setting, one can consider a **basis** of functions $b_1, \ldots, b_n$, $n \leq m$ (e.g. polynomials of degree $\leq n-1$) and the model

$$y = c_1 b_1(x) + c_2 b_2(x) + \cdots + c_n b_n(x),$$

obtaining then $A = A_{(ij)} = b_j(t_i)$ and $\boldsymbol{c} = (c_1, \ldots, c_n)^\mathsf{T}$.
The resulting overdetermined linear system is $m \times n$, and we assume A to have full column rank.
In the following, we describe some approaches to solve such linear system in a least-squares sense.

Normal equations

Let us recall and rewrite our minimization problem as

$$\min_{\boldsymbol{c}\in\mathbb{R}^n} \frac{1}{2}\boldsymbol{r}^\intercal\boldsymbol{r} = \frac{1}{2}(A\boldsymbol{c}-\boldsymbol{y})^\intercal(A\boldsymbol{c}-\boldsymbol{y}) = R(\boldsymbol{c}).$$

We have

$$R(\boldsymbol{c}) = \frac{1}{2}((A\boldsymbol{c})^\intercal A\boldsymbol{c} - (A\boldsymbol{c})^\intercal\boldsymbol{y} - \boldsymbol{y}^\intercal A\boldsymbol{c} + \boldsymbol{y}^\intercal\boldsymbol{y}).$$

Note that $(A\boldsymbol{c})^\intercal\boldsymbol{y} = \boldsymbol{y}^\intercal A\boldsymbol{c}$ (it is a scalar product), therefore

$$R(\boldsymbol{c}) = \frac{1}{2}((A\boldsymbol{c})^\intercal A\boldsymbol{c} - 2(A\boldsymbol{c})^\intercal\boldsymbol{y} + \boldsymbol{y}^\intercal\boldsymbol{y}).$$

Normal equations (cont.)

At this point, we can use the fact that if $\boldsymbol{c}$ is a minimum for $R(\boldsymbol{c})$ then $\frac{\partial R}{\partial \boldsymbol{c}} = 0$. We then calculate

$$\frac{\partial R}{\partial \boldsymbol{c}} = A^\mathsf{T} A\boldsymbol{c} - A^\mathsf{T}\boldsymbol{y} = 0,$$

and thus the **normal equations**

$$A^\mathsf{T} A\boldsymbol{c} = A^\mathsf{T}\boldsymbol{y}.$$

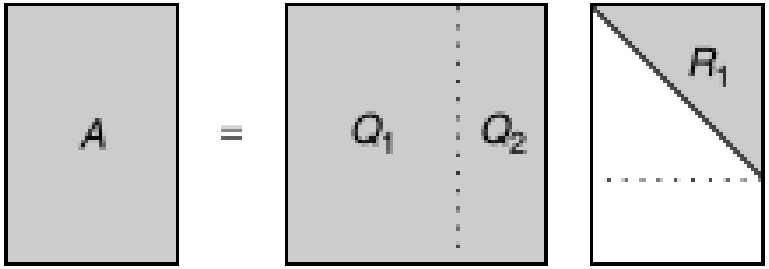
We point out that $A^\mathsf{T} A$ is a $n \times n$ positive definite matrix, therefore we can use e.g. Cholesky factorization to find $\boldsymbol{c}$. This approach is dangerous from a numerical perspective if A is severely ill-conditioned, since in that case multiplying by A^T may amplify too much the ill-conditioning in the linear system.

QR decomposition

Any $m \times n$ matrix A can be decomposed as $A = QR$, where Q is an $m \times m$ orthogonal matrix $Q^{-1} = Q^{\mathsf{T}}$ and R is $m \times n$ so that

$$R = \begin{bmatrix} R_1 \\ 0 \end{bmatrix},$$

where R_1 is a $n \times n$ upper triangular matrix. If A is square, then R is itself square and upper triangular. We can highlight the submatrix Q_2 that multiplies the zero part of R.



QR decomposition (cont.)

The QR factorization can be employed as a direct method for solving square linear systems. Here, we use it to find the least-squares solution as follows. First, since the multiplication by an orthogonal matrix does not modify the length of a vector, we can write

$$R(\boldsymbol{c}) = \frac{1}{2}(A\boldsymbol{c} - \boldsymbol{y})^\intercal(A\boldsymbol{c} - \boldsymbol{y}) = \frac{1}{2}\|A\boldsymbol{c} - \boldsymbol{y}\|_2^2 = \frac{1}{2}\|Q^\intercal(QR\boldsymbol{c} - \boldsymbol{y})\|_2^2.$$

Then, also omitting the $1/2$ factor (no need here), we have

$$\|Q^\intercal(QR\boldsymbol{c} - \boldsymbol{y})\|_2^2 = \left\| \begin{bmatrix} R_1 \\ 0 \end{bmatrix} \boldsymbol{c} - \begin{bmatrix} Q_1^\intercal \\ Q_2^\intercal \end{bmatrix} \boldsymbol{y} \right\|_2^2 = \|R_1\boldsymbol{c} - Q_1^\intercal\boldsymbol{y}\|_2^2 + \|Q_2^\intercal\boldsymbol{y}\|_2^2.$$

QR decomposition (cont.)

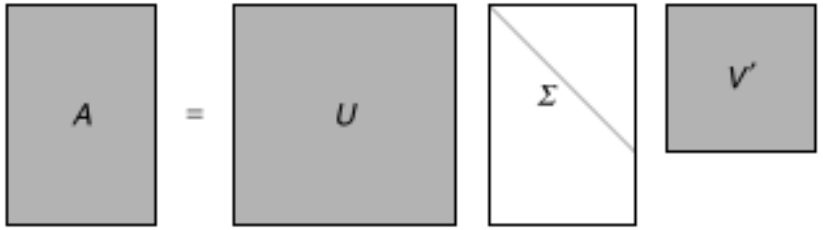
The least-squares solution can thus be found by solving

$$R_1 \boldsymbol{c} = Q_1^\mathsf{T} \boldsymbol{y},$$

while $\|Q_2^\mathsf{T} \boldsymbol{y}\|_2^2$ is the residual.
The QR decomposition is a stable approach, but computationally is not that cheap, being $\mathcal{O}(m^2 n)$.
We observe that if A is square, then Q_2 is the zero matrix, and there is no residual.

Singular Value Decomposition (SVD)

Similarly to the QR factorization, the SVD is a direct method for solving linear systems, in an exact or least-squares sense.
Any $m \times n$ matrix A can be decomposed as $A = U\Sigma V^{\mathsf{T}}$, where U and V are $m \times m$ and $n \times n$ orthogonal matrices, respectively, and Σ is composed by a diagonal $n \times n$ matrix, with a zero $(m-n) \times n$ matrix below. The diagonal elements $\sigma_1, \ldots, \sigma_n$ are the so-called **singular values**.



Singular Value Decomposition (SVD) (cont.)

To find the solution of the original linear system, similar steps with respect to the QR approach can be carried out. Here, we point out a different formulation.

The least-squares solution of $A\boldsymbol{c} = \boldsymbol{y}$ can be expressed as

$$\boldsymbol{c} = V\Sigma^{+}U^{\mathsf{T}}\boldsymbol{y},$$

where Σ^{+} is the **pseudoinverse** of Σ, which is

$$\Sigma = \begin{pmatrix} \sigma_1 & & \\ & \ddots & \\ & & \sigma_n \\ 0 & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & 0 \end{pmatrix} \Rightarrow \Sigma^{+} = \begin{pmatrix} 1/\sigma_1 & & & 0 & \dots & 0 \\ & \ddots & & \vdots & \ddots & \vdots \\ & & 1/\sigma_n & 0 & \dots & 0 \end{pmatrix}.$$

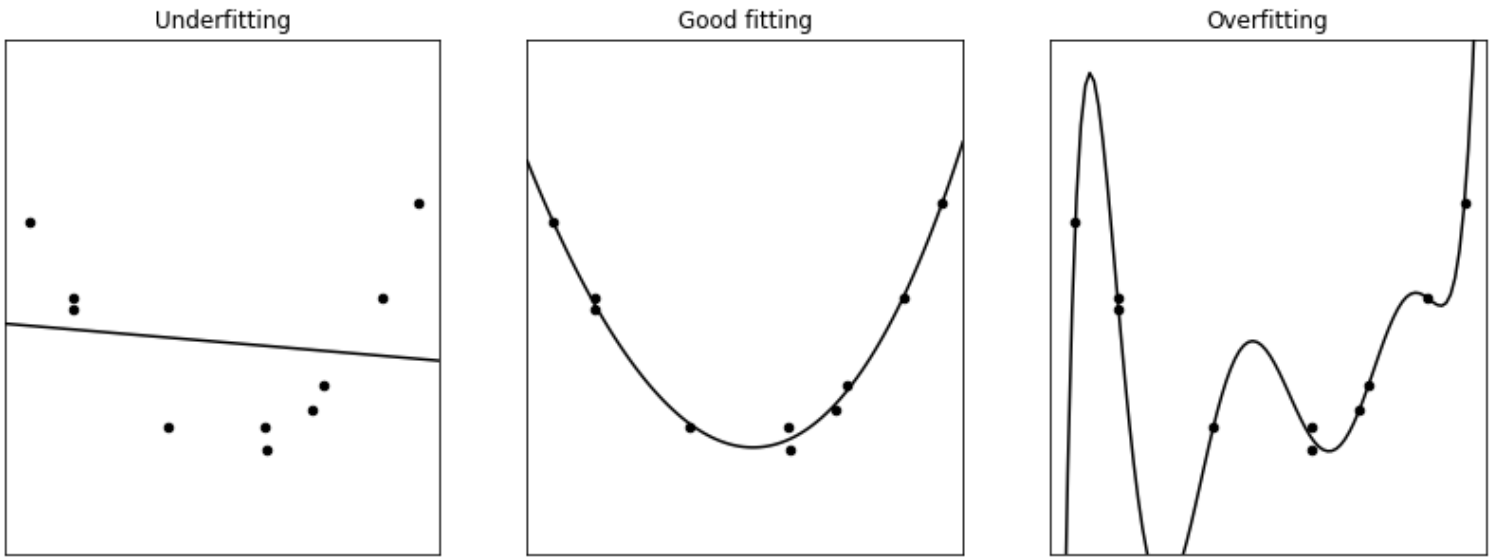
Some final remarks

The SVD actually works in the more general case where A is not of full rank, and it also reveals more properties than the QR decomposition. However, this comes with a slightly larger computational cost.

When the number of basis functions coincide with the number of observations ($m = n$) we talk about **interpolation**. In this case, it is possible for the model to exactly replicate all the observations $y_1, \ldots, y_n$ at the sampling times $t_1, \ldots, t_n$. This could be required, depending on the application.

On model complexity

How to choose n? How much complexity should we include in our model?
This is a crucial issue. Especially in the machine learning field, we talk about **underfitting** and **overfitting**.



Approximating integrals

In numerical integration, one wants to find a *good* approximation of

$$\int_a^b f(x)\mathrm{d}x \approx \sum_{i=1}^{n} w_i f(x_i), \tag{1}$$

where $x_1, \ldots, x_n$ are nodes in $[a, b]$ and $w_1, \ldots, w_n$ are weights. In the following, we provide a possible formal approach for the construction of such weights, restricting to the case of *equidistant* nodes. In this framework, we will call the resulting quadrature schemes **Newton-Cotes** rules. Their property is being **exact** for polynomials up to a certain degree, i.e., there is an equal sign in (1). To simplify the discussion, we restrict to the cases $n = 2, 3$.

Trapezoidal rule ($n = 2$)

The nodes are $x_1 = a$, $x_2 = b$. We impose the exactness on polynomials of degree ≤ 1 by writing

$$\begin{cases} w_1 x_1^0 + w_2 x_2^0 = \int_a^b 1 \mathrm{d}x = b - a, \\ w_1 x_1^1 + w_2 x_2^1 = \int_a^b x \mathrm{d}x = \frac{b^2 - a^2}{2}. \end{cases}$$

Therefore, we get

$$\begin{cases} w_1 + w_2 = b - a, \\ a w_1 + b w_2 = \frac{b^2 - a^2}{2}. \end{cases}$$

The solution is $w_1 = w_2 = \frac{b-a}{2}$.

Simpson's rule ($n = 3$)

The nodes are $x_1 = a$, $x_2 = (a+b)/2$ and $x_3 = b$. We impose the exactness on polynomials of degree ≤ 2 by writing

$$\begin{cases} w_1 x_1^0 + w_2 x_2^0 + w_3 x_3^0 = \int_a^b 1 \mathrm{d}x = b - a, \\ w_1 x_1^1 + w_2 x_2^1 + w_3 x_3^1 = \int_a^b x \mathrm{d}x = \frac{b^2 - a^2}{2}. \\ w_1 x_1^2 + w_2 x_2^2 + w_3 x_3^2 = \int_a^b x^2 \mathrm{d}x = \frac{b^3 - a^3}{3}. \end{cases}$$

Therefore, we get

$$\begin{cases} w_1 + w_2 + w_3 = b - a, \\ a w_1 + \frac{b-a}{2} w_2 + b w_3 = \frac{b^2 - a^2}{2}. \\ a^2 w_1 + \frac{(b-a)^2}{4} w_2 + b^2 w_3 = \frac{b^3 - a^3}{3}. \end{cases}$$

The solution is $w_1 = w_3 = \frac{b-a}{6}$, $w_2 = \frac{2(b-a)}{3}$. Simpson's rule is actually exact for polynomials up to degree 3 (!!!)

Composite rules

It is not convenient to use Newton-Cotes rules that are exact for high degree polynomials, because they are not stable. Instead, it is common to employ **composite** rules, i.e., to divide $[a, b]$ into subintervals and apply the quadrature rules locally on each subinterval

- In the trapezoidal case, we need two nodes for each subinterval.
- In the Simpson's case, we need three nodes for each subinterval.