Linear Regression

Guillaume Frèche

Version 1.0

Keywords: linear regression, least squares, recursive least squares, autoregressive signals.

In this document, we present the simplest model of regression: **linear regression**. We introduce the model, we derive **least squares** exact solution and **recursive least squares** (RLS) algorithm, we extend to some variants of the model and we apply these results to **autoregressive signals**.

1 Problem presentation

Suppose that we are given $n \in \mathbb{N}^*$ vectors $\mathbf{x}_i = (x_{i,1}, \dots, x_{i,m})^\mathsf{T} \in \mathbb{R}^m$ and n scalars $y_1, \dots, y_n \in \mathbb{R}$. We aim at determining weights $w_1, \dots, w_m$ such that for any $i \in [\![1, n]\!]$,

$$y_i = \sum_{j=1}^{m} w_j x_{i,j} = \mathbf{x}_i^\mathsf{T} \mathbf{w}_m$$

This problem is called **linear regression**. It can be rewritten matricially as

$$\mathbf{y}_n = \mathrm{X}_{n,m}\mathbf{w}_m \qquad \text{where} \qquad \mathbf{y}_n = \begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix} \qquad \mathrm{X}_{n,m} = \begin{pmatrix} x_{1,1} & \cdots & x_{1,m} \\ \vdots & \ddots & \vdots \\ x_{n,1} & \cdots & x_{n,m} \end{pmatrix} = \begin{pmatrix} \mathbf{x}_1^\mathsf{T} \\ \vdots \\ \mathbf{x}_n^\mathsf{T} \end{pmatrix} \qquad \mathbf{w}_m = \begin{pmatrix} w_1 \\ \vdots \\ w_m \end{pmatrix}$$

Now this is a linear system with data $\mathbf{y}_n$ and $\mathrm{X}_{n,m}$, and unknown $\mathbf{w}_m$. There are three possible cases:

- If $n < m$, the linear system is **underdetermined** and admits an infinity of solutions $\mathbf{w}_m$. This case will be treated in an upcoming document.
- If $n = m$, the linear system is **determined**. If $\det(\mathrm{X}_{m,m}) \neq 0$, i.e. if matrix $\mathrm{X}_{m,m}$ is invertible, then the solution is given by $\mathbf{w}_m = \mathrm{X}_{m,m}^{-1}\mathbf{y}_m$. If $\det(\mathrm{X}_{m,m}) = 0$, at least one row of this matrix is linearly dependent on the others, and the corresponding equations can be removed from the system, which boils down to the underdetermined case.
- If $n > m$, the linear system is **overdetermined**, meaning that it does not necessarily have an exact solution. In this document, we are interested in this case.

2 Estimation perspective

If $\mathbf{y}_n$ was obtained by computing $\mathrm{X}_{n,m}\mathbf{w}_m$, then the linear regression problem should have an exact solution: $\mathbf{w}_m$, even for $n > m$. However, if $\mathbf{y}_n$ is a noisy version of $\mathrm{X}_{n,m}\mathbf{w}_m$, there might not be an exact solution. We denote $\hat{\mathbf{y}}_n(\mathrm{X}_{n,m}, \mathbf{w}_m) =$

$\mathrm{X}_{n,m}\mathbf{w}_m$ the exact noiseless output and

$$\mathbf{y}_n = \hat{\mathbf{y}}_n(\mathrm{X}_{n,m}, \mathbf{w}_m) + \mathbf{e}_n = \mathrm{X}_{n,m}\mathbf{w}_m + \mathbf{e}_n \tag{1}$$

the noisy observed output, where $\mathbf{e}_n = (e_1, \dots, e_n)^\mathsf{T}$ is a vector of n i.i.d realizations of some random variable E. We focus here on the case where E is a zero-mean random variable following a normal distribution $E \sim \mathcal{N}(0, \sigma_e^2)$ for some error variance $\sigma_e^2 > 0$. In this case, $\mathbf{y}_n$ is a realization of random vector $\mathbf{Y}_n(\mathrm{X}_{n,m}, \mathbf{w}_m)$ following distribution $\mathcal{N}\left(\hat{\mathbf{y}}_n(\mathrm{X}_{n,m}, \mathbf{w}_m), \Sigma\right)$, where $\Sigma = \sigma_e^2 \mathrm{I}_n$. The principle of **parametric estimation** is to provide an estimator $\widehat{\mathbf{w}}_m$ of the unknown weights $\mathbf{w}_m$ based on the optimization of some functional depending on observed data $\mathbf{y}_n$ and $\mathrm{X}_{n,m}$, and on hidden weights $\mathbf{w}_m$. A commonly used functional is the **likelihood** defined as:

$$\mathcal{L}(\mathbf{y}_n, \mathrm{X}_{n,m}, \mathbf{w}_m) = f_{\mathbf{Y}_n(\mathrm{X}_{n,m}, \mathbf{w}_m)}(\mathbf{y}_n)$$

that is, the probability density of random vector $\mathbf{Y}_n(\mathrm{X}_{n,m}, \mathbf{w}_m)$ outcome being $\mathbf{y}_n$. The corresponding estimator, called the **maximum likelihood estimator**, is given by:

$$\widehat{\mathbf{w}}_m = \operatorname*{arg\,max}_{\mathbf{w}_m \in \mathbb{R}^m} \left(\mathcal{L}(\mathbf{y}_n, \mathrm{X}_{n,m}, \mathbf{w}_m)\right) = \operatorname*{arg\,max}_{\mathbf{w}_m \in \mathbb{R}^m} \left(f_{\mathbf{Y}_n(\mathrm{X}_{n,m}, \mathbf{w}_m)}(\mathbf{y}_n)\right)$$

For some distributions f, such as the normal distribution that we develop later, it is more interesting to compute the **log likelihood**:

$$\lambda(\mathbf{y}_n, \mathrm{X}_{n,m}, \mathbf{w}_m) = \ln \mathcal{L}(\mathbf{y}_n, \mathrm{X}_{n,m}, \mathbf{w}_m) = \ln f_{\mathbf{Y}_n(\mathrm{X}_{n,m}, \mathbf{w}_m)}(\mathbf{y}_n)$$

Since logarithm is monotically increasing, maximum likelihood and maximum log likelihood estimators are exactly the same. The latter one is given by:

$$\widehat{\mathbf{w}}_m = \operatorname*{arg\,max}_{\mathbf{w}_m \in \mathbb{R}^m} \left(\lambda(\mathbf{y}_n, \mathrm{X}_{n,m}, \mathbf{w}_m)\right) = \operatorname*{arg\,max}_{\mathbf{w}_m \in \mathbb{R}^m} \left(\ln f_{\mathbf{Y}_n(\mathrm{X}_{n,m}, \mathbf{w}_m)}(\mathbf{y}_n)\right)$$

Now let us compute this maximum log likelihood estimator in the case of additive gaussian noise.

$$\begin{aligned}
\ln f_{\mathbf{Y}_n(\mathrm{X}_{n,m}, \mathbf{w}_m)}(\mathbf{y}_n) &= \ln \left(\frac{1}{(2\pi)^{\frac{n}{2}} \sqrt{\det(\Sigma)}} \exp\left(-\frac{1}{2} (\mathbf{y}_n - \hat{\mathbf{y}}_n(\mathrm{X}_{n,m}, \mathbf{w}_m))^\mathsf{T} \Sigma^{-1} (\mathbf{y}_n - \hat{\mathbf{y}}_n(\mathrm{X}_{n,m}, \mathbf{w}_m)) \right) \right) \\
&= -\frac{n}{2} \ln(2\pi) - n \ln(\sigma_e) - \frac{1}{2\sigma_e^2} \left\| \mathbf{y}_n - \hat{\mathbf{y}}_n(\mathrm{X}_{n,m}, \mathbf{w}_m) \right\|^2
\end{aligned}$$

where we consider the ℓ_2-norm on $\mathbb{R}^n$:

$$\forall \mathbf{v} = (v_1, \dots, v_n)^\mathsf{T} \in \mathbb{R}^n \qquad \|\mathbf{v}\|^2 = \sum_{i=1}^n v_i^2 = \mathbf{v}^\mathsf{T}\mathbf{v}$$

Since noise variance σ_e^2 is a fixed parameter of the problem, we see that maximizing log likelihood is equivalent to minimizing the ℓ_2-distance between noisy observation $\mathbf{y}_n$ and noiseless outcome $\hat{\mathbf{y}}_n(\mathrm{X}_{n,m}, \mathbf{w}_m) = \mathrm{X}_{n,m}\mathbf{w}_m$. This subsequent problem is known as **least squares**.

3 Least squares exact solution

Our minimization problem is now:

$$\widehat{\mathbf{w}}_m = \operatorname*{arg\,min}_{\mathbf{v}_m \in \mathbb{R}^m} \|\mathbf{y}_n - \mathrm{X}_{n,m}\mathbf{v}_m\|^2$$

Since $\ell : \mathbf{v} \mapsto \mathbf{y_n} - \mathrm{X}_{n,m}\mathbf{v}$ is linear and $\psi : \mathbf{v} \mapsto \|\mathbf{v}\|^2$ is convex, mapping $\varphi = \psi \circ \ell : \mathbf{v} \mapsto \|\mathbf{y_n} - \mathrm{X}_{n,m}\mathbf{v}\|^2$ is convex and admits a unique global minimum in $\widehat{\mathbf{w}}_m$ on $\mathbb{R}^m$. To find the minimum, we compute the derivative φ with respect to $\mathbf{v}$ using the chain rule:

$$\frac{\partial \varphi}{\partial \mathbf{v}} = \frac{\partial \ell}{\partial \mathbf{v}} \frac{\partial \psi}{\partial \ell}$$

On one hand,

$$\frac{\partial \psi}{\partial \ell} = \left(\frac{\partial \psi}{\partial \ell_1}, \dots, \frac{\partial \psi}{\partial \ell_n} \right)^\mathsf{T} = (2\ell_1, \dots, 2\ell_n)^\mathsf{T} = 2\ell$$

On the other hand,

$$\frac{\partial \ell}{\partial \mathbf{v}} = \begin{pmatrix} \frac{\partial \ell_1}{\partial v_1} & \cdots & \frac{\partial \ell_n}{\partial v_1} \\ \vdots & \ddots & \vdots \\ \frac{\partial \ell_1}{\partial v_m} & \cdots & \frac{\partial \ell_n}{\partial v_m} \end{pmatrix}$$

For any $i \in [\![1, m]\!]$ and any $j \in [\![1, n]\!]$,

$$\frac{\partial \ell_j}{\partial v_i} = \frac{\partial}{\partial v_i} \left(y_j - (x_{j,1}, \dots, x_{j,m})(v_1, \dots, v_m)^\mathsf{T} \right) = -x_{j,i}$$

which implies $\frac{\partial \ell}{\partial \mathbf{v}} = -\mathrm{X}_{n,m}^\mathsf{T}$ and $\frac{\partial \varphi}{\partial \mathbf{v}} = -2\mathrm{X}_{n,m}^\mathsf{T} \left(\mathbf{y_n} - \mathrm{X}_{n,m}\mathbf{v} \right)$. Thus we have

$$\frac{\partial \varphi}{\partial \mathbf{v}}(\widehat{\mathbf{w}}_m) = -2\mathrm{X}_{n,m}^\mathsf{T}\mathbf{y_n} + 2\mathrm{X}_{n,m}^\mathsf{T}\mathrm{X}_{n,m}\widehat{\mathbf{w}}_m = 0$$

Finally, we find a direct expression of least squares estimator $\widehat{\mathbf{w}}_m$ as a function of data $\mathrm{X}_{n,m}$ and $\mathbf{y}_n$:

$$\widehat{\mathbf{w}}_m = \left(\mathrm{X}_{n,m}^\mathsf{T}\mathrm{X}_{n,m} \right)^{-1} \mathrm{X}_{n,m}^\mathsf{T}\mathbf{y}_n \tag{2}$$

Remark: It would be tempting to simplify this expression by writing $\left(\mathrm{X}_{n,m}^\mathsf{T}\mathrm{X}_{n,m} \right)^{-1} = \mathrm{X}_{n,m}^{-1} \left(\mathrm{X}_{n,m}^\mathsf{T} \right)^{-1}$, but we have to keep in mind that, since $m > n$, we are dealing with rectangular matrices $\mathrm{X}_{n,m}$ which might not have well-defined inverses. Therefore, we will stick with Equation (2).

4 Recursive least squares

Equation (2) gives the expression of the weights knowning the n samples $\mathbf{x}_i$ and y_i at once. With the **recursive least squares** (RLS) algorithm, we can update the weights with new data on the fly. Imagine that we have already found the least squares estimator $\widehat{\mathbf{w}}_m^{(k)}$ for $\mathbf{y}_k = \mathrm{X}_{k,m}\mathbf{w}_m^{(k)}$ and we want to use this estimator to solve the least squares problem $\mathbf{y}_{k+1} = \mathrm{X}_{k+1,m}\mathbf{w}_m^{(k+1)}$. From Equation (2), we have

$$\widehat{\mathbf{w}}_m^{(k+1)} = \left(\mathrm{X}_{k+1,m}^\mathsf{T}\mathrm{X}_{k+1,m} \right)^{-1} \mathrm{X}_{k+1,m}^\mathsf{T}\mathbf{y}_{k+1} = \left(\mathrm{X}_{k,m}^\mathsf{T}\mathrm{X}_{k,m} + \mathbf{x}_{k+1}\mathbf{x}_{k+1}^\mathsf{T} \right)^{-1} \left(\mathrm{X}_{k,m}^\mathsf{T}\mathbf{y}_k + \mathbf{x}_{k+1}y_{k+1} \right) \tag{3}$$

We need the following inversion lemma to go further:

Lemma 4.1

- **Woodbury matrix identity**: If $\mathrm{A} \in \mathbb{R}^p$, $\mathrm{B} \in \mathbb{R}^{p \times q}$, $\mathrm{C} \in \mathbb{R}^p$ and $\mathrm{D} \in \mathbb{R}^{q \times p}$ are four matrices where A and C are invertible, then

$$(\mathsf{A}+\mathsf{BCD})^{-1} = \mathsf{A}^{-1} - \mathsf{A}^{-1}\mathsf{B}(\mathsf{C}^{-1}+\mathsf{DA}^{-1}\mathsf{B})^{-1}\mathsf{DA}^{-1}$$

- **Sherman-Morrison formula**: in particular, if $\mathbf{b} \in \mathbb{R}^p$ and $\mathbf{d} \in \mathbb{R}^p$ are vectors,

$$(\mathsf{A}+\mathbf{b}\mathbf{d}^\mathsf{T})^{-1} = \mathsf{A}^{-1} - \frac{\mathsf{A}^{-1}\mathbf{b}\mathbf{d}^\mathsf{T}\mathsf{A}^{-1}}{1+\mathbf{d}^\mathsf{T}\mathsf{A}^{-1}\mathbf{b}} \tag{4}$$

Note that denominator $1+\mathbf{d}^\mathsf{T}\mathsf{A}^{-1}\mathbf{b}$ is a scalar.

Proof : We have

$$\begin{aligned}
&(\mathsf{A}+\mathsf{BCD})(\mathsf{A}^{-1} - \mathsf{A}^{-1}\mathsf{B}(\mathsf{C}^{-1}+\mathsf{DA}^{-1}\mathsf{B})^{-1}\mathsf{DA}^{-1}) \\
&\quad = \mathsf{I}_p + \mathsf{BCDA}^{-1} - \mathsf{B}(\mathsf{C}^{-1}+\mathsf{DA}^{-1}\mathsf{B})^{-1}\mathsf{DA}^{-1} - \mathsf{BCDA}^{-1}\mathsf{B}(\mathsf{C}^{-1}+\mathsf{DA}^{-1}\mathsf{B})^{-1}\mathsf{DA}^{-1} \\
&\quad = \mathsf{I}_p + \mathsf{BCDA}^{-1} - \left[\mathsf{B}(\mathsf{C}^{-1}+\mathsf{DA}^{-1}\mathsf{B})^{-1}\mathsf{DA}^{-1} + \mathsf{BCDA}^{-1}\mathsf{B}(\mathsf{C}^{-1}+\mathsf{DA}^{-1}\mathsf{B})^{-1}\mathsf{DA}^{-1}\right] \\
&\quad = \mathsf{I}_p + \mathsf{BCDA}^{-1} - \left[\mathsf{B}+\mathsf{BCDA}^{-1}\mathsf{B}\right](\mathsf{C}^{-1}+\mathsf{DA}^{-1}\mathsf{B})^{-1}\mathsf{DA}^{-1} \\
&\quad = \mathsf{I}_p + \mathsf{BCDA}^{-1} - \mathsf{BC}\left[\mathsf{C}^{-1}+\mathsf{DA}^{-1}\mathsf{B}\right](\mathsf{C}^{-1}+\mathsf{DA}^{-1}\mathsf{B})^{-1}\mathsf{DA}^{-1} \\
&\quad = \mathsf{I}_p + \mathsf{BCDA}^{-1} - \mathsf{BCDA}^{-1} = \mathsf{I}_p
\end{aligned}$$

We get the same result if we change the order of the two factors, thereby proving Woodbury matrix identity. We obtain the Sherman-Morrison formula by setting $\mathsf{B}=\mathbf{b}$ and $\mathsf{D}=\mathbf{d}^\mathsf{T}$ as vectors, and $\mathsf{C}=1$ as a scalar. ■

We apply this lemma with $\mathsf{A}=\mathsf{X}_{k,m}^\mathsf{T}\mathsf{X}_{k,m}$ and $\mathbf{b}=\mathbf{d}=\mathbf{x}_{k+1}$. This gives

$$\widehat{\mathbf{w}}_m^{(k+1)} = \left((\mathsf{X}_{k,m}^\mathsf{T}\mathsf{X}_{k,m})^{-1} - \frac{(\mathsf{X}_{k,m}^\mathsf{T}\mathsf{X}_{k,m})^{-1}\mathbf{x}_{k+1}\mathbf{x}_{k+1}^\mathsf{T}(\mathsf{X}_{k,m}^\mathsf{T}\mathsf{X}_{k,m})^{-1}}{1+\mathbf{x}_{k+1}^\mathsf{T}(\mathsf{X}_{k,m}^\mathsf{T}\mathsf{X}_{k,m})^{-1}\mathbf{x}_{k+1}}\right)\left(\mathsf{X}_{k,m}^\mathsf{T}\mathbf{y}_k + \mathbf{x}_{k+1}y_{k+1}\right)$$

Setting $\mathsf{P}_k = (\mathsf{X}_{k,m}^\mathsf{T}\mathsf{X}_{k,m})^{-1}$, we can develop:

$$\begin{aligned}
\widehat{\mathbf{w}}_m^{(k+1)} &= \mathsf{P}_k\mathsf{X}_{k,m}^\mathsf{T}\mathbf{y}_k + \mathsf{P}_k\mathbf{x}_{k+1}y_{k+1} - \frac{\mathsf{P}_k\mathbf{x}_{k+1}\mathbf{x}_{k+1}^\mathsf{T}\mathsf{P}_k}{1+\mathbf{x}_{k+1}^\mathsf{T}\mathsf{P}_k\mathbf{x}_{k+1}}\left(\mathsf{X}_{k,m}^\mathsf{T}\mathbf{y}_k + \mathbf{x}_{k+1}y_{k+1}\right) \\
&= \widehat{\mathbf{w}}_m^{(k)} + \frac{\mathsf{P}_k\mathbf{x}_{k+1}\left(1+\mathbf{x}_{k+1}^\mathsf{T}\mathsf{P}_k\mathbf{x}_{k+1}\right)y_{k+1}}{1+\mathbf{x}_{k+1}^\mathsf{T}\mathsf{P}_k\mathbf{x}_{k+1}} - \frac{\mathsf{P}_k\mathbf{x}_{k+1}\mathbf{x}_{k+1}^\mathsf{T}\mathsf{P}_k\left(\mathsf{X}_{k,m}^\mathsf{T}\mathbf{y}_k + \mathbf{x}_{k+1}y_{k+1}\right)}{1+\mathbf{x}_{k+1}^\mathsf{T}\mathsf{P}_k\mathbf{x}_{k+1}} \\
&= \widehat{\mathbf{w}}_m^{(k)} + \frac{\mathsf{P}_k\mathbf{x}_{k+1}\left(y_{k+1} - \mathbf{x}_{k+1}^\mathsf{T}\mathsf{P}_k\mathsf{X}_{k,m}^\mathsf{T}\mathbf{y}_k\right)}{1+\mathbf{x}_{k+1}^\mathsf{T}\mathsf{P}_k\mathbf{x}_{k+1}} \\
&= \widehat{\mathbf{w}}_m^{(k)} + \frac{\mathsf{P}_k\mathbf{x}_{k+1}}{1+\mathbf{x}_{k+1}^\mathsf{T}\mathsf{P}_k\mathbf{x}_{k+1}}\left(y_{k+1} - \mathbf{x}_{k+1}^\mathsf{T}\widehat{\mathbf{w}}_m^{(k)}\right)
\end{aligned}$$

Finally, we rewrite this expression as:

$$\widehat{\mathbf{w}}_m^{(k+1)} = \widehat{\mathbf{w}}_m^{(k)} + \mathbf{g}_{k+1}\left(y_{k+1} - \mathbf{x}_{k+1}^\mathsf{T}\widehat{\mathbf{w}}_m^{(k)}\right) \tag{5}$$

where

$$\mathbf{g}_{k+1} = \frac{(\mathsf{X}_{k,m}^\mathsf{T}\mathsf{X}_{k,m})^{-1}\mathbf{x}_{k+1}}{1+\mathbf{x}_{k+1}^\mathsf{T}(\mathsf{X}_{k,m}^\mathsf{T}\mathsf{X}_{k,m})^{-1}\mathbf{x}_{k+1}} = \frac{\mathsf{P}_k\mathbf{x}_{k+1}}{1+\mathbf{x}_{k+1}^\mathsf{T}\mathsf{P}_k\mathbf{x}_{k+1}} \in \mathbb{R}^m \tag{6}$$

This latter term can be considered as a special form of Kalman gain, that we will detail in an upcoming document. The major issue in computing $\mathbf{g}_{k+1}$ is to determine the inverse matrix $\mathsf{P}_k = \left(\mathsf{X}_{k,m}^{\mathsf{T}}\mathsf{X}_{k,m}\right)^{-1}$. We have

$$\mathsf{P}_{k+1} = (\mathsf{X}_{k+1,m}^{\mathsf{T}}\mathsf{X}_{k+1,m})^{-1} = (\mathsf{X}_{k,m}^{\mathsf{T}}\mathsf{X}_{k,m} + \mathbf{x}_{k+1}\mathbf{x}_{k+1}^{\mathsf{T}})^{-1}$$

Using again the inversion lemma,

$$\mathsf{P}_{k+1} = \mathsf{P}_k - \frac{\mathsf{P}_k\mathbf{x}_{k+1}\mathbf{x}_{k+1}^{\mathsf{T}}\mathsf{P}_k}{1 + \mathbf{x}_{k+1}^{\mathsf{T}}\mathsf{P}_k\mathbf{x}_{k+1}}$$

Using the definition (6) of $\mathbf{g}_{k+1}$, we get:

$$\mathsf{P}_{k+1} = \mathsf{P}_k - \mathbf{g}_{k+1}\mathbf{x}_{k+1}^{\mathsf{T}}\mathsf{P}_k = (\mathsf{I}_m - \mathbf{g}_{k+1}\mathbf{x}_{k+1}^{\mathsf{T}})\mathsf{P}_k \tag{7}$$

We can wrap up identities (5), (6) and (7) into the following algorithm. With no prior information on data, it is common to initialize the algorithm with $\widehat{\mathbf{w}}_m^{(0)} = 0$ and $P_0 = \delta\mathsf{I}_m$, where $\delta \ll \sigma_e^2$.

Algorithm 1 Recursive least squares

```
1: procedure RLS($\mathsf{X}_{n,m}$, $\mathbf{y}$)
2:     Input $\mathbf{x}_i = (x_{i,1}, \dots, x_{i,m})$ for $i \in [\![1, n]\!]$
3:     Input $y_i$ for $i \in [\![1, n]\!]$
4:     Set $\widehat{\mathbf{w}}_m^{(0)} = 0$
5:     Set $\mathsf{P}_0 = \delta\mathsf{I}_m$
6:     for $k \in [\![1, n]\!]$ do
7:         $\mathbf{g}_k \leftarrow \dfrac{\mathsf{P}_{k-1}\mathbf{x}_k}{1 + \mathbf{x}_k^{\mathsf{T}}\mathsf{P}_{k-1}\mathbf{x}_k}$
8:         $\widehat{\mathbf{w}}_m^{(k)} \leftarrow \widehat{\mathbf{w}}_m^{(k-1)} + \mathbf{g}_k\left(y_k - \mathbf{x}_k^{\mathsf{T}}\widehat{\mathbf{w}}_m^{(k-1)}\right)$
9:         $\mathsf{P}_k \leftarrow (\mathsf{I}_m - \mathbf{g}_k\mathbf{x}_k^{\mathsf{T}})\mathsf{P}_{k-1}$
10:    end for
11:    Return $\widehat{\mathbf{w}}_m^{(n)}$
12: end procedure
```

5 Variants

We study some variants of linear regression, where we only need to adapt the definition of the data matrix $\mathsf{X}_{n,m}$ without changing the RLS algorithm to update the weights.

5.1 Affine regression

Suppose that we are given $n \in \mathbb{N}^*$ vectors $\mathbf{x}_i = (x_{i,1}, \dots, x_{i,m})^{\mathsf{T}} \in \mathbb{R}^m$ and n scalars $y_1, \dots, y_n$ and we are trying to find weights $w_1, \dots, w_m$ and bias w_0 such that for any $i \in [\![1, n]\!]$,

$$y_i = \sum_{j=1}^{m} w_j x_{i,j} + w_0$$

This expression can be seen as the inner product in $\mathbb{R}^{m+1}$ of vectors $(1, x_{i,1}, \dots, x_{i,m})$ and $(w_0, w_1, \dots, w_m)$, yielding to:

$$\mathrm{X}_{n,m+1} = \begin{pmatrix} 1 & x_{1,1} & \cdots & x_{1,m} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_{n,1} & \cdots & x_{n,m} \end{pmatrix} \quad \text{and} \quad \mathbf{w}_{m+1} = \begin{pmatrix} w_0 \\ w_1 \\ \vdots \\ w_m \end{pmatrix}$$

5.2 Polynomial regression

Suppose that we are given n distinct scalar input x_i yielding to the polynomial ouput:

$$\forall i \in [\![1, n]\!] \qquad y_i = P(x_i) = w_0 + w_1 x_i + \cdots + w_m x_i^m = \mathbf{x}_i^\mathsf{T} \mathbf{w}_{m+1} \qquad \text{where} \qquad \mathbf{x}_i = (1, x_i, \dots, x_i^m)^\mathsf{T}$$

This is a particular case of affine regression where $x_{i,j} = x_i^j$. We can write:

$$\mathrm{X}_{n,m+1} = \begin{pmatrix} 1 & x_1 & \cdots & x_1^m \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_n & \cdots & x_n^m \end{pmatrix} \quad \text{and} \quad \mathbf{w}_{m+1} = \begin{pmatrix} w_0 \\ w_1 \\ \vdots \\ w_m \end{pmatrix}$$

Remark: When $n = m + 1$, the system is determined and known as **Lagrange interpolation**. Its exact solution is

$$P(x) = \sum_{i=1}^{n} y_i \ell_i(x)$$

where ℓ_i is the i-th Lagrange polynomial

$$\ell_i(x) = \prod_{\substack{j \in [\![1,n]\!] \\ j \neq i}} \frac{x - x_j}{x_i - x_j}$$

such that $\ell_i(x_i) = 1$ and $\ell_i(x_j) = 0$ for $j \neq i$.

5.3 Comparison of affine regression, quadratic regression and Lagrange interpolation

We are given a black box producing a noisy quadratic function:

$$y = a(x - \alpha)^2 + b + e = w_0 + w_1 x + w_2 x^2 + e$$

where e is a realization of some additive zero-mean gaussian noise $E \sim \mathcal{N}(0, \sigma^2)$. We propose three models for this simulation: affine regression, quadratic regression (i.e. polynomial of order 2) and Lagrange interpolation. We train our models on $n_{\text{train}} = 20$ samples and test them on $n_{\text{test}} = 20$ samples. Figure 1 compares the performance of the 3 models. For each regression method and for both training and test datasets, we computed the mean square error between the ouput generated by the black box and the ouput produced by the three models.

	Affine regression	Quadratic regression	Lagrange interpolation
Training error	63.835	2.8658×10^{-2}	7.0271×10^{-15}
Test error	67.922	5.5889×10^{-2}	9.6124×10^{7}

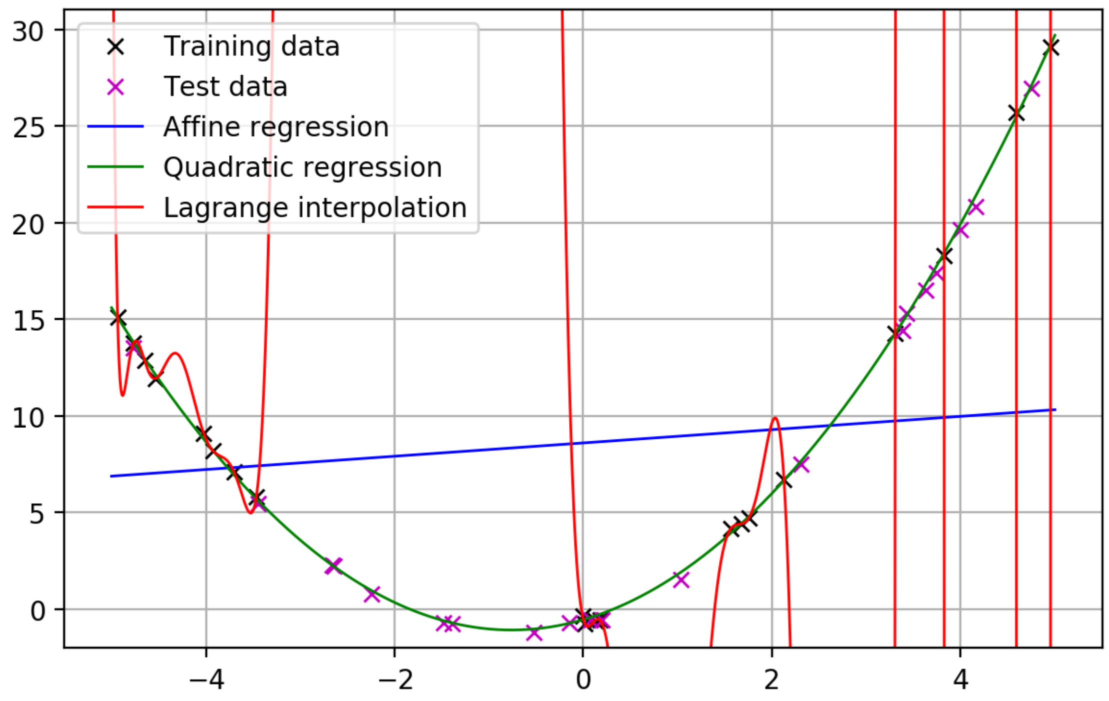

Figure 1: Performance comparison of the 3 regression models

First, note that errors of affine and quadratic regression are of the same order of magnitude on both training and test datasets, with the quadratic regression exhibiting better performance than the affine one, which should not be surprising since the hidden function is quadratic. On the other hand, Lagrange interpolation is excellent on training, its deviation from 0 being due to computational approximations, while it performs very poorly on the test dataset. This phenomenon is called **overfitting**. When we choose a model, we fix the **hypothesis space**, i.e. a space of functions from which we are going to pick the mapping fitting best our datasets. For affine regression, this is the space $\mathcal{P}_1$ of polynomials of degree at most 1, for quadratic regression , the space $\mathcal{P}_2$ of polynomials of degree at most 2, which includes $\mathcal{P}_1$, and for Lagrange interpolation, the space $\mathcal{P}_{n_{\text{train}}}$ of polynomials of degree at most n_{train}, including both previous spaces. With affine regression, the set is too small and the RLS algorithm has difficulties finding a close estimate. This issue is called **underfitting** and is usually solved by expanding the hypothesis space, as we do by changing $\mathcal{P}_1$ into $\mathcal{P}_2$ switching from affine to quadratic regression. With Lagrange interpolation and overfitting, the hypothesis space is too large, and although the algorithm works very well during training, it has difficulties **generalizing** to new data.

5.4 Bilinear regression

Now suppose that we are given $n \in \mathbb{N}^*$ vectors $\mathbf{x}_i = (x_{i,1}, \dots, x_{i,m}) \in \mathbb{R}^m$ and n scalars $y_1, \dots, y_n$ and we are trying to find a weight matrix $\mathrm{Q}_m = (q_{j,k})$ such that for any $i \in [\![1, n]\!]$,

$$y_i = \sum_{j=1}^{m} \sum_{k=1}^{m} q_{j,k} x_{i,j} x_{i,k} = \mathbf{x}_i^\mathsf{T} \mathrm{Q}_m \mathbf{x}_i$$

By commutativity of the product $x_{i,j}x_{i,k} = x_{i,k}x_{i,j}$, we can have two matrices Q_m and R_m such that $q_{j,k} + q_{k,j} = r_{j,k} + r_{k,j}$, and $\mathbf{x}_i^\mathsf{T} \mathrm{Q}_m \mathbf{x}_i = \mathbf{x}_i^\mathsf{T} \mathrm{R}_m \mathbf{x}_i$, thus matrix Q_m is not unique. To impose unicity, we set Q_m to be an upper triangular matrix. To have a form similar to the previous ones, we flatten matrix Q_m:

$$y_i = \tilde{\mathbf{x}}_i^\mathsf{T} \widetilde{Q}_m \qquad \text{with} \qquad \begin{cases} \tilde{\mathbf{x}}_i &= (x_{i,1}^2, \quad x_{i,1}x_{i,2}, \quad \dots, \quad x_{i,1}x_{i,m}, \quad x_{i,2}^2, \quad \dots, \quad x_{i,j}x_{i,k}, \quad \dots, \quad x_{i,m}^2)^\mathsf{T} \\ \widetilde{Q}_m &= (q_{1,1}, \quad q_{1,2}, \quad \dots, \quad q_{1,m}, \quad q_{2,2}, \quad \dots, \quad q_{i,j}, \quad \dots, \quad q_{m,m})^\mathsf{T} \end{cases}$$

6 The autoregressive model

6.1 Presentation

A discrete-time signal $(z_k)_{k\in\mathbb{N}}$ is **autoregressive** of order m if its samples satisfy a recursive relation of the form:

$$\forall k \in \mathbb{N} \quad k \geq m \qquad z_k = \sum_{i=1}^{m} w_i z_{k-i}$$

If z is of finite length $n > m$, we can deduce $n - m$ linear regression relations

$$y_k = \mathbf{x}_k^\mathsf{T}\mathbf{w}_m \qquad \text{where} \qquad y_k = z_k \qquad \text{and} \qquad \mathbf{x}_k^\mathsf{T} = (z_{k-1}, \dots, z_{k-m})$$

hence the name autoregressive.

6.2 Weighted RLS

For this kind of signals, hidden weights $\mathbf{w}_m$ may vary over time and while applying recursive least squares, we want to give less importance to previous samples as they are are further in the past. This is why we can modify Equation (3) to introduce a **forgetting factor** $\lambda \in [0, 1]$:

$$\widehat{\mathbf{w}}_m^{(k+1)} = \left(\lambda X_{k,m}^\mathsf{T} X_{k,m} + \mathbf{x}_{k+1}\mathbf{x}_{k+1}^\mathsf{T}\right)^{-1} \left(\lambda X_{k,m}^\mathsf{T}\mathbf{y}_k + \mathbf{x}_{k+1}y_{k+1}\right)$$

We get

$$\widehat{\mathbf{w}}_m^{(k+1)} = \widehat{\mathbf{w}}_m^{(k)} + \mathbf{g}_{k+1}\left(y_{k+1} - \mathbf{x}_{k+1}^\mathsf{T}\widehat{\mathbf{w}}_m^{(k)}\right)$$

where

$$\mathbf{g}_{k+1} = \frac{P_k\mathbf{x}_{k+1}}{\lambda + \mathbf{x}_{k+1}^\mathsf{T} P_k \mathbf{x}_{k+1}} \qquad \text{and} \qquad P_{k+1} = \frac{1}{\lambda}(I_m - \mathbf{g}_{k+1}\mathbf{x}_{k+1}^\mathsf{T})P_k$$

If $\lambda = 1$, we retrieve regular RLS described earlier.

6.3 Application to speech processing

The autoregressive model is a good way to appromixate speech signals. The estimated autoregressive coefficients can yield to many applications such as speech recognition and transcription, word classification or speaker identification. In this example, we applied the weighted RLS algorithm on a speech signal representing the word ”Hello” with forgetting factors $\lambda = 1$, $\lambda = 0.92$ and $\lambda = 0.89$. Figure 2 displays the speech signal over time. Figures 3 shows the evolution of estimated weights for $\lambda = 1$ and $\lambda = 0.89$. We notice that for $\lambda = 1$, weigths are rapidly fixed. Since it corresponds to the regular RLS algorithm, weights should provide the best fitting on each of the 26,460 samples in the speech signal. It shows that providing new samples towards the end have little influence on the estimation process, like when we estimate the mean of a large dataset. Finally, Figure 4 displays the evolution of estimation error on a logarithmic scale for forgetting factors $\lambda = 1$,

$\lambda = 0.92$ and $\lambda = 0.89$. We can notice that if the average error does not improve much for λ between 0.92 and 1, it rapidly decreases for λ below 0.90.

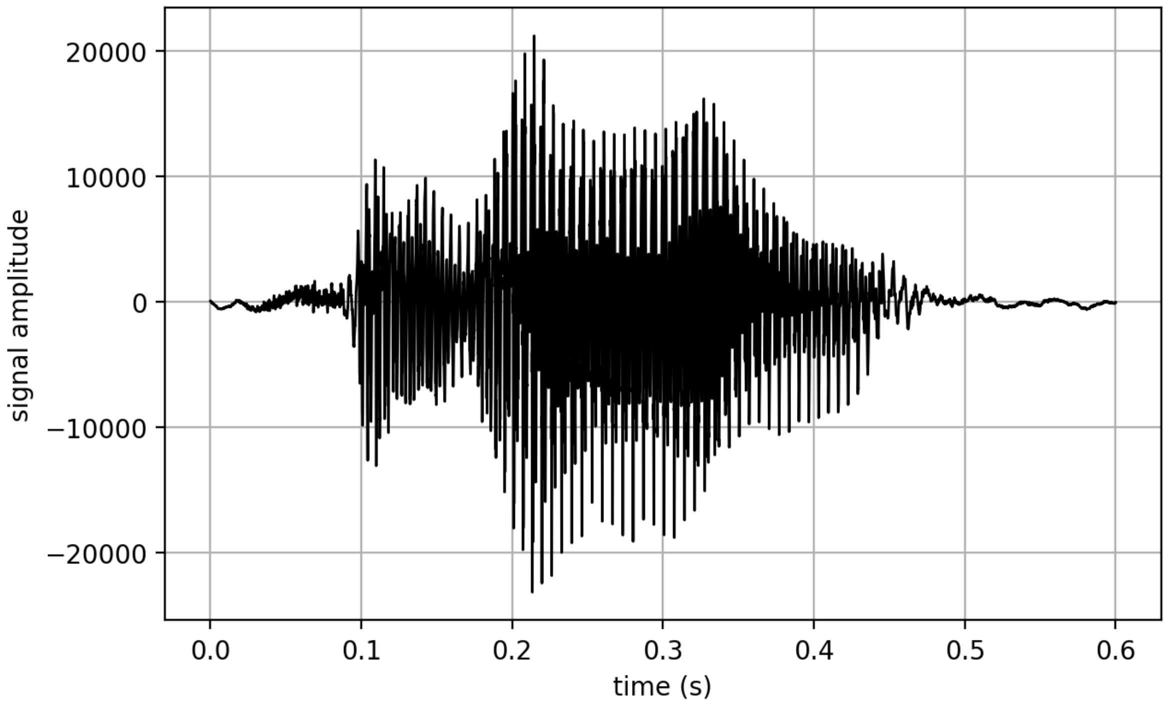

Figure 2: Speech signal of the word ”Hello”

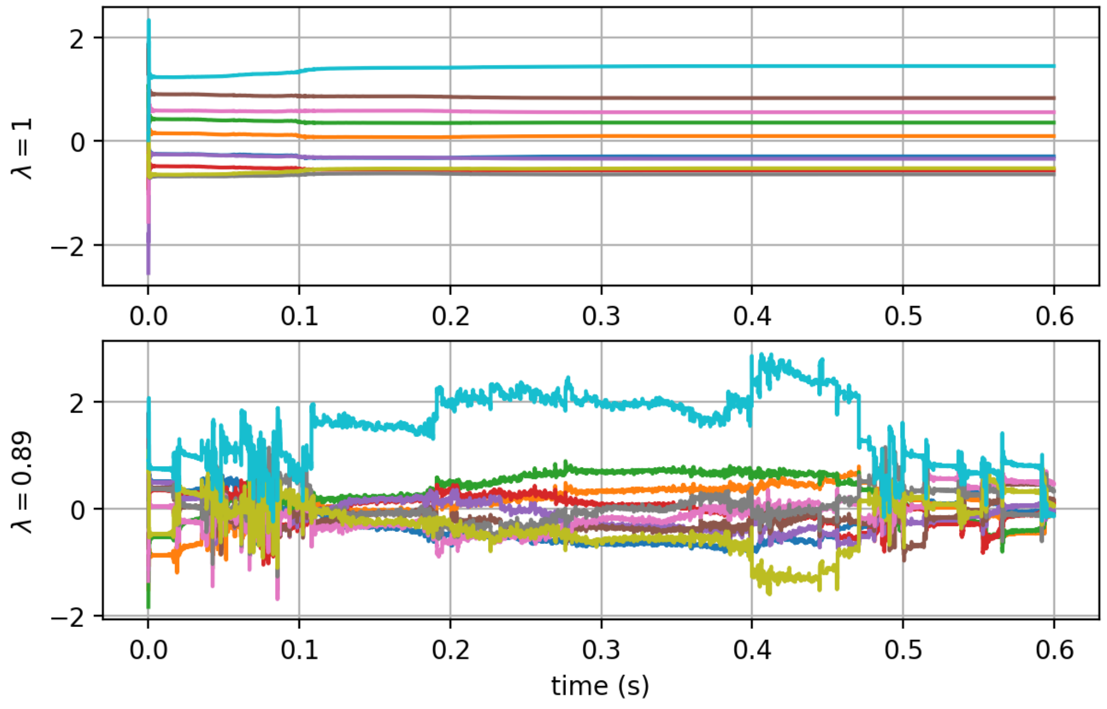

Figure 3: Evolution of weights estimated by RLS algorithm with forgetting factors $\lambda = 1$ and $\lambda = 0.89$

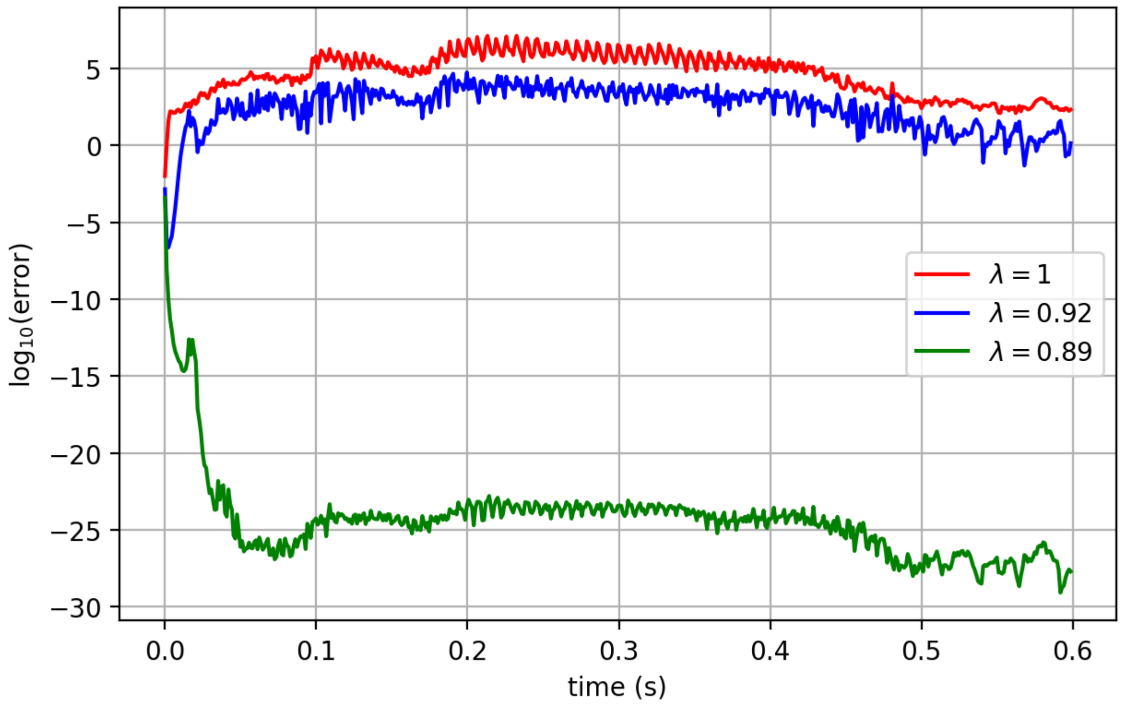

Figure 4: Evolution of the estimation error on a logarithmic scale