UNIVERSITÉ DE LIÈGE

MASTER IN ENGINEERING

November 17, 2020

ELEN 0062
Introduction to Machine Learning

Project 2

Bias and variance analysis

Maxime Lione - **s175606**
Cédric Siboyabasore - **s175202**

1 Analytical derivations

1.1 *Bayes model and residual error in classification*

1.1. (a) *Analytical formulation of the Bayes model corresponding to the zero-one error loss.*

The zero-one error loss function is given by

$$L(h_b(x_0,x_1), y) = I(h_b(x_0,x_1) \neq y) = \begin{cases} 1 & \text{if } h_b(x_0,x_1) \neq y \\ 0 & \text{if } h_b(x_0,x_1) = y \end{cases}$$

where I is the indicator function.
The Bayes model $h_b(x_0,x_1)$ is such that

$$\begin{aligned} h_b(x_0,x_1) &= \arg\max_{y^i} P(y = y^i \mid x_0,x_1) \\ &= \arg\max_{y^i} \frac{P(x_0,x_1 \mid y = y^i)P(y = y^i)}{P(x_0,x_1)} && \text{by Bayes's theorem} \\ &= \arg\max_{y^i} P(x_0,x_1 \mid y = y^i)P(y = y^i) && \text{since } P(x_0,x_1) \text{ is independent of } y^i \end{aligned}$$

Since the class $y^i \in \{-1,+1\}$ is uniformly selected then $P(y = y^i) = \frac{1}{2}$
We have

$$\begin{aligned} h_b(x_0,x_1) &= \arg\max_{y^i} P(x_0,x_1 \mid y = y^i) * \frac{1}{2} \\ &= \arg\max_{y^i} P(x_0,x_1 \mid y = y^i) \end{aligned}$$

The probability on the sample $\boldsymbol{x^i} = (x_0^i, x_1^i)$ values or $\mathbf{x^i} = [x_0^i, x_1^i]^T$ in matrix notation given its class y^i is given by

$$\begin{aligned} P(x_0^i, x_1^i \mid y = y^i) = \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma}_i) &= \mathcal{N}\left(\begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 & \rho^i \\ \rho^i & 1 \end{bmatrix} \right) \\ &= \frac{1}{\sqrt{(2\pi)^2 |\boldsymbol{\Sigma}_i|}} \exp\left\{ -\frac{1}{2} \left(\mathbf{x^i} - \boldsymbol{\mu}\right)^T \boldsymbol{\Sigma}_i^{-1} \left(\mathbf{x^i} - \boldsymbol{\mu}\right) \right\} \\ &= \frac{1}{(4\pi^2)^{1/2} |\boldsymbol{\Sigma}_i|^{1/2}} \exp\left\{ -\frac{1}{2} \left(\mathbf{x^i} - \boldsymbol{\mu}\right)^T \Sigma_i^{-1} \left(\mathbf{x^i} - \boldsymbol{\mu}\right) \right\} \\ &= (4\pi^2 |\boldsymbol{\Sigma}_i|)^{-1/2} \exp\left\{ -\frac{1}{2} \left(\mathbf{x^i} - \boldsymbol{\mu}\right)^T \Sigma_i^{-1} \left(\mathbf{x^i} - \boldsymbol{\mu}\right) \right\} \end{aligned}$$

With a Bayes model we classify to the most probable class so we have $h_b(x_0,x_1) = 1$ if

$$P(x_0,x_1 \mid y = +1) > P(x_0,x_1 \mid y = -1)$$

$$\iff (4\pi^2 |\boldsymbol{\Sigma}_1|)^{-1/2} \exp\left\{ -\frac{1}{2} (\mathbf{x} - \boldsymbol{\mu})^T \boldsymbol{\Sigma}_1^{-1} (\mathbf{x} - \boldsymbol{\mu}) \right\} > (4\pi^2 |\boldsymbol{\Sigma}_{-1}|)^{-1/2} \exp\left\{ -\frac{1}{2} (\mathbf{x} - \boldsymbol{\mu})^T \Sigma_{-1}^{-1} (\mathbf{x} - \boldsymbol{\mu}) \right\}$$

Since the logarithmic function *ln* is strictly increasing, we can apply this function on both members of the inequality

$$\rightarrow -\frac{1}{2}ln(4\pi^2) - \frac{1}{2}ln(|\mathbf{\Sigma}_1|) - \frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^T \Sigma_1^{-1}(\mathbf{x}-\boldsymbol{\mu}) > -\frac{1}{2}ln(4\pi^2) - \frac{1}{2}ln(|\mathbf{\Sigma}_{-1}|) - \frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^T \mathbf{\Sigma}_{-1}^{-1}(\mathbf{x}-\boldsymbol{\mu})$$

$$\iff -\frac{1}{2}ln(|\mathbf{\Sigma}_1|) - \frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^T \Sigma_1^{-1}(\mathbf{x}-\boldsymbol{\mu}) > -\frac{1}{2}ln(|\mathbf{\Sigma}_{-1}|) - \frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^T \Sigma_{-1}^{-1}(\mathbf{x}-\boldsymbol{\mu})$$

$$\iff -\frac{1}{2}ln(\begin{vmatrix} 1 & \rho^+ \\ \rho^+ & 1 \end{vmatrix}) - \frac{1}{2}\left(\begin{bmatrix} x_0 \\ x_1 \end{bmatrix} - \begin{bmatrix} 0 \\ 0 \end{bmatrix}\right)^T \begin{bmatrix} 1 & \rho^+ \\ \rho^+ & 1 \end{bmatrix}^{-1}\left(\begin{bmatrix} x_0 \\ x_1 \end{bmatrix} - \begin{bmatrix} 0 \\ 0 \end{bmatrix}\right)$$

$$> -\frac{1}{2}ln(\begin{vmatrix} 1 & \rho^- \\ \rho^- & 1 \end{vmatrix}) - \frac{1}{2}\left(\begin{bmatrix} x_0 \\ x_1 \end{bmatrix} - \begin{bmatrix} 0 \\ 0 \end{bmatrix}\right)^T \begin{bmatrix} 1 & \rho^- \\ \rho^- & 1 \end{bmatrix}^{-1}\left(\begin{bmatrix} x_0 \\ x_1 \end{bmatrix} - \begin{bmatrix} 0 \\ 0 \end{bmatrix}\right)$$

$$\iff -\frac{1}{2}ln(1-(\rho^+)^2) - \frac{1}{2}\begin{bmatrix} x_0 & x_1 \end{bmatrix} \frac{1}{1-(\rho^+)^2} \begin{bmatrix} 1 & -\rho^+ \\ -\rho^+ & 1 \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \end{bmatrix}$$

$$> -\frac{1}{2}ln(1-(\rho^-)^2) - \frac{1}{2}\begin{bmatrix} x_0 & x_1 \end{bmatrix} \frac{1}{1-(\rho^-)^2} \begin{bmatrix} 1 & -\rho^- \\ -\rho^- & 1 \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \end{bmatrix}$$

Since $\rho^- = -\rho^+$

$$-\frac{1}{2*(1-(\rho^+)^2)}\begin{bmatrix} x_0 & x_1 \end{bmatrix} \begin{bmatrix} 1 & -\rho^+ \\ -\rho^+ & 1 \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \end{bmatrix} > -\frac{1}{2*(1-(\rho^+)^2)}\begin{bmatrix} x_0 & x_1 \end{bmatrix} \begin{bmatrix} 1 & \rho^+ \\ \rho^+ & 1 \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \end{bmatrix}$$

$$\iff -\frac{1}{2*(1-(\rho^+)^2)}(x_0^2 - 2\rho^+ x_0 x_1 + x_1^2) > -\frac{1}{2*(1-(\rho^+)^2)}(x_0^2 + 2\rho^+ x_0 x_1 + x_1^2)$$

$$\iff -\frac{1}{(1-(\rho^+)^2)}(x_0^2 - 2\rho^+ x_0 x_1 + x_1^2) > -\frac{1}{(1-(\rho^+)^2)}(x_0^2 + 2\rho^+ x_0 x_1 + x_1^2)$$

According to Sylvester's criterion, since the covariance matrix $\mathbf{\Sigma}_i$ is positive semi-definite its determinant (which is a principal minor of the matrix) must be nonnegative.
Hence $|\mathbf{\Sigma}_1| = |\mathbf{\Sigma}_2| = \frac{1}{(1-(\rho^+)^2)}$ is positive. We can simplify this term in both members of the inequation without changing the inequality direction :

$$-(x_0^2 - 2\rho^+ x_0 x_1 + x_1^2) > -(x_0^2 + 2\rho^+ x_0 x_1 + x_1^2)$$
$$\iff x_0^2 - 2\rho^+ x_0 x_1 + x_1^2 < x_0^2 + 2\rho^+ x_0 x_1 + x_1^2$$
$$\iff 4\rho^+ x_0 x_1 > 0$$
$$\iff x_0 x_1 > 0 \quad \text{because } \rho^+ > 0$$

We find that the Bayes model $h_b(x0, x1)$ is given by

$$h_b(x0, x1) = \begin{cases} +1 & \text{if } x_0 x_1 > 0 \\ -1 & \text{otherwise} \end{cases}$$

1.1. (b) *Analytical formulation and estimation of the residual error*

Script Q1_b.py

The generalization error, which is the prediction error of our model $h_b(x0,x1)$ on new data, is the probability of predicting the wrong output. It is given by :

$$\begin{aligned} E_{x_0,x_1,y}\{1(y \neq h_b(x_0,x_1)\} &= P(y \neq h_b(x_0,x_1)) \\ &= \sum_{y_i} P(y = y_i \cap h_b(x_0,x_1) \neq y_i) \\ &= P(h_b(x_0,x_1) = +1 \cap y = -1) + P(h_b(x_0,x_1) = -1 \cap y = +1) \end{aligned}$$

With conditional probability properties, we have :

$$\begin{aligned} &E_{x_0,x_1,y}\{1(y \neq h_b(x_0,x_1)\} \\ &\quad = P(h_b(x_0,x_1) = -1|\, y = +1)P(y = +1) + P(h_b(x_0,x_1) = +1|y = -1)P(y = -1) \end{aligned}$$

Since we know that $P(y = +1) = P(y = -1) = \frac{1}{2}$, we have :

$$\begin{aligned} &E_{x_0,x_1,y}\{1(y \neq h_b(x_0,x_1)\} \\ &\quad = \frac{1}{2}\Big(P(h_b(x_0,x_1) = -1|\, y = +1) + P(h_b(x_0,x_1) = +1|y = -1)\Big) \end{aligned}$$

Let's compute the two probabilities :

• We had found that our Bayes model $h_b(x0,x1) = +1$ if $x_0x_1 > 0$, thus we have :

$$\begin{aligned} P(h_b(x_0,x_1) = +1|y = -1) &= \iint_{x0x1>0} \frac{1}{\sqrt{(2\pi)^2\,|\boldsymbol{\Sigma}_{-1}|}} \exp\left\{-\frac{1}{2}\left(\mathbf{x}-\boldsymbol{\mu}\right)^T \boldsymbol{\Sigma}_{-1}^{-1}\left(\mathbf{x}-\boldsymbol{\mu}\right)\right\} \mathrm{d}x_0\mathrm{d}x_1 \\ &= \iint_{x0x1>0} \frac{1}{2\pi\sqrt{1-(\rho^+)^2}} \exp\left\{-\frac{1}{2*(1-(\rho^+)^2)}(x_0^2 + 2\rho^+ x_0x_1 + x_1^2)\right\} \mathrm{d}x_0\mathrm{d}x_1 \\ &= \iint_{x0x1>0} \frac{1}{2\pi\sqrt{1-(\rho^+)^2}} \exp\left\{-\frac{1}{2*(1-(\rho^+)^2)}(x_0^2 + 2\rho^+ x_0x_1 + x_1^2)\right\} \mathrm{d}x_0\mathrm{d}x_1 \\ &= \int_{-\infty}^{0}\int_{-\infty}^{0} \frac{1}{2\pi\sqrt{1-(\rho^+)^2}} \exp\left\{-\frac{1}{2*(1-(\rho^+)^2)}(x_0^2 + 2\rho^+ x_0x_1 + x_1^2)\right\} \mathrm{d}x_0\mathrm{d}x_1 \\ &+ \int_{0}^{+\infty}\int_{0}^{+\infty} \frac{1}{2\pi\sqrt{1-(\rho^+)^2}} \exp\left\{-\frac{1}{2*(1-(\rho^+)^2)}(x_0^2 + 2\rho^+ x_0x_1 + x_1^2)\right\} \mathrm{d}x_0\mathrm{d}x_1 \end{aligned}$$

and we had found that $h_b(x0,x1) = -1$ otherwise, meaning if $x_0x_1 < 0$. We thus have :

$$\begin{aligned}
P(h_b(x_0,x_1) = -1|\, y = +1) &= \iint_{x0x1<0} \frac{1}{\sqrt{(2\pi)^2\,|\boldsymbol{\Sigma}_1|}} \exp\left\{-\frac{1}{2}\,(\mathbf{x}-\boldsymbol{\mu})^T\,\boldsymbol{\Sigma}_1^{-1}\,(\mathbf{x}-\boldsymbol{\mu})\right\} \mathrm{d}x_0\mathrm{d}x_1 \\
&= \iint_{x0x1<0} \frac{1}{2\pi\sqrt{1-(\rho^+)^2}} \exp\left\{-\frac{1}{2*(1-(\rho^+)^2)}(x_0^2 - 2\rho^+ x_0x_1 + x_1^2)\right\} \mathrm{d}x_0\mathrm{d}x_1 \\
&= \int_{-\infty}^{0}\int_{0}^{\infty} \frac{1}{2\pi\sqrt{1-(\rho^+)^2}} \exp\left\{-\frac{1}{2*(1-(\rho^+)^2)}(x_0^2 - 2\rho^+ x_0x_1 + x_1^2)\right\} \mathrm{d}x_0\mathrm{d}x_1 \\
&+ \int_{0}^{+\infty}\int_{-\infty}^{0} \frac{1}{2\pi\sqrt{1-(\rho^+)^2}} \exp\left\{-\frac{1}{2*(1-(\rho^+)^2)}(x_0^2 - 2\rho^+ x_0x_1 + x_1^2)\right\} \mathrm{d}x_0\mathrm{d}x_1
\end{aligned}$$

We find $P(h_b(x_0,x_1) = +1|y = -1) = P(h_b(x_0,x_1) = -1|y = +1) = 0.230053$ thanks to Python *nquad* function that allows us to compute integrals.
We eventually have

$$\begin{aligned}
&E_{x_0,x_1,y}\{1(y \neq h_b(x_0,x_1))\} \\
&\quad = \frac{1}{2}\Big(P(h_b(x_0,x_1) = -1|\, y = +1) + P(h_b(x_0,x_1) = +1|y = -1)\Big) \\
&\quad = 0.230053
\end{aligned}$$

It means the minimal attainable error rate for this classification problem is of 23.0053%.
We verified our estimation of the residual error in the same script :
we first uniformly generate $N = 3000$ y^i samples with either value -1 or $+1$ using Python *np.random.choice* function. They correspond to the classes.We initialize a *hb* vector that will correspond to our Bayes model.
We then loop over all y^i values that we stored in a vector y. If at iteration i we have $y^i = 1$ then the current ρ value is 0.75 and the covariance matrix is $\boldsymbol{\Sigma}_1 = \begin{bmatrix} 1 & 0.75 \\ 0.75 & 1 \end{bmatrix}$. Otherwise, $\rho = -1$ and $\boldsymbol{\Sigma}_{-1} = \begin{bmatrix} 1 & -0.75 \\ -0.75 & 1 \end{bmatrix}$
Given the class y^i we then draw the sample values x_0^i and x_0^i from the multivariate normal distribution with our covariance matrix and mean matrix $\boldsymbol{\mu} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$ thanks to the *np.random.multivariate_normal* Python function.

If $x_0^i x_1^i > 0$, our current Bayes model $hb^i = h_b(x_0^i, x_1^i) = 1$ otherwise $hb^i = h_b(x_0^i, x_1^i) = -1$.
We eventually compare our h_b and y vectors and obtain the empirical residual error by computing the mean of errors which happen when $hb^i \neq y^i$.

We find an empirical residual error of 0.231213 which supports our estimation because it is close to the residual error $E_{x_0,x_1,y}\{1\,(y \neq h_b(x_0,x_1))\} = 0.230053$ we computed analytically.

1.2 *Bias and variance of ridge regression*

1.2. (a)

When we use the ordinary least-square method we try to minimize the error

$$\sum_{i=1}^{N} \left(y_i - \boldsymbol{x}_i^T \boldsymbol{w}\right)^2$$

In order to find weight $\boldsymbol{w} = \boldsymbol{w}_{OLS}$ we differentiate the error with respect to $\boldsymbol{w}$ and we get

$$-2\sum_{i=1}^{N} \boldsymbol{x}_i^T \left(y_i - \boldsymbol{x}_i^T \boldsymbol{w}\right) = 0$$

$$\iff \sum_{i=1}^{N} \boldsymbol{x}_i^T \left(y_i - \boldsymbol{x}_i^T \boldsymbol{w}\right) = 0$$

In matrix form the equation becomes

$$\mathbf{X}^T(\mathbf{y} - \mathbf{X}\boldsymbol{w}) = 0$$

where $\mathbf{X}$ is a $n x p$ matrix, $\mathbf{y}$ is a $n x 1$ vector and $\boldsymbol{w}$ is a $p x 1$ vector.

$$\iff \mathbf{X}^T\mathbf{y} - \mathbf{X}^T\mathbf{X}\boldsymbol{w} = 0$$
$$\iff \mathbf{X}^T\mathbf{X}\boldsymbol{w} = \mathbf{X}^T\mathbf{y}$$
$$\iff \boldsymbol{w} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y} = \boldsymbol{w}_{OLS}$$

Since $\mathbf{X}$ is orthogonal we have

$$\boldsymbol{w}_{OLS} = \mathbf{X}^T\mathbf{y}$$

When using the ridge regression method we try to minimize the error

$$\sum_{i=1}^{N} \left(y_i - \boldsymbol{x}_i^T \boldsymbol{w}\right)^2 + \lambda \boldsymbol{w}^T \boldsymbol{w}$$

In order to find ridge regression weight $\boldsymbol{w} = \boldsymbol{w}_R$ we differentiate the error with respect to $\boldsymbol{w}$ and we get

$$-2\sum_{i=1}^{N} \boldsymbol{x}_i^T \left(y_i - \boldsymbol{x}_i^T \boldsymbol{w}\right) + 2\lambda \boldsymbol{w} = 0$$

since $\frac{\partial \lambda \boldsymbol{w}^T \boldsymbol{w}}{\partial \boldsymbol{w}} = 2\lambda \boldsymbol{w}$. We can write

$$-\sum_{i=1}^{N} \boldsymbol{x}_i^T \left(y_i - \boldsymbol{x}_i^T \boldsymbol{w}\right) + \lambda \boldsymbol{w} = 0$$

and in matrix notation

$$-\mathbf{X}^T(\mathbf{y} - \mathbf{X}\boldsymbol{w}) + \lambda \boldsymbol{w} = 0$$

$$\iff -\mathbf{X}^T\mathbf{y} + \mathbf{X}^T\mathbf{X}\boldsymbol{w} + \lambda \boldsymbol{w} = 0$$

$$\iff (\mathbf{X}^T\mathbf{X}+\lambda\mathbf{I})\boldsymbol{w} = \mathbf{X}^T\mathbf{y}$$
$$\iff \boldsymbol{w} = (\mathbf{X}^T\mathbf{X}+\lambda\mathbf{I})^{-1}\mathbf{X}^T\mathbf{y} = \boldsymbol{w}_R$$

Since $\mathbf{X}$ is orthogonal we have

$$\begin{aligned}\boldsymbol{w}_R &= (\mathbf{I}+\lambda\mathbf{I})^{-1}\mathbf{X}^T\mathbf{y}\\ &= (1+\lambda)^{-1}\mathbf{I}\mathbf{X}^T\mathbf{y}\\ &= (1+\lambda)^{-1}\mathbf{X}^T\mathbf{y}\end{aligned}$$

Injecting $\boldsymbol{w}_{OLS} = \mathbf{X}^T\mathbf{y}$ in the expression of $\boldsymbol{w}_R$, we get

$$\boldsymbol{w}_R = \frac{\boldsymbol{w}_{OLS}}{1+\lambda}$$

1.2. (b)

Now, let's assume that our model is :

$$\hat{y}(\boldsymbol{x}) = \frac{\boldsymbol{x}^T \boldsymbol{w}_{OLS}}{1+\lambda}$$

with :

$$w_{OLS} = \arg\min_{\boldsymbol{w}\in\mathbb{R}^p} \sum_{i=1}^{N} (y_i - \boldsymbol{x}_i^T \boldsymbol{w})^2$$

In order to find the bias and variance of our model, let's start from the expression of its error. As our model is a function of several inputs, its error can be written as :

$$E_{\boldsymbol{x},y}\left\{(y-\hat{y}(\boldsymbol{x}))^2\right\}$$

When averaging over all learning sets, we get :

$$\begin{aligned} E &= E_{LS}\left\{E_{\boldsymbol{x},y}\left\{(y-\hat{y}(\boldsymbol{x}))^2\right\}\right\} \\ &= E_{\boldsymbol{x}}\left\{E_{LS}\left\{E_{y|\boldsymbol{x}}\left\{(y-\hat{y}(\boldsymbol{x}))^2\right\}\right\}\right\} \\ &= E_{\boldsymbol{x}}\left\{var_{y|\boldsymbol{x}}\{y\}\right\} + E_{\boldsymbol{x}}\left\{bias^2(\boldsymbol{x})\right\} + E_{\boldsymbol{x}}\left\{var_{LS}\{\hat{y}(\boldsymbol{x})\}\right\} \\ &= noise(\boldsymbol{x}) + bias^2(\boldsymbol{x}) + variance(\boldsymbol{x}) \end{aligned}$$

Where :

- $noise(\boldsymbol{x}) = E_{y|\boldsymbol{x}}\left\{(y - E_{y|\boldsymbol{x}}\{y\})^2\right\}$
- $bias^2(\boldsymbol{x}) = (E_{y|\boldsymbol{x}}\{y\} - E_{LS}\{\hat{y}(\boldsymbol{x})\})^2$
- $variance(\boldsymbol{x}) = E_{LS}\left\{(\hat{y}(\boldsymbol{x}) - E_{LS}\{\hat{y}(\boldsymbol{x})\})^2\right\}$

because $h_B(\boldsymbol{x}) = E_{y|\boldsymbol{x}}\{y\}$.

Furthermore, as we consider $y = f(\boldsymbol{x}) + \varepsilon$ where $\varepsilon \sim \mathcal{N}(0,\sigma^2)$, we can write :

$$\begin{aligned} E_{y|\boldsymbol{x}}\{y\} &= E\{f(\boldsymbol{x}) + \varepsilon\} \\ &= E\{f(\boldsymbol{x})\} + E\{\varepsilon\} \\ &= f(\boldsymbol{x}) \end{aligned}$$

because given $\boldsymbol{x}$, $f(\boldsymbol{x})$ is a constant thus its expectation is equal to itself.

- When $\lambda = 0$, we have :

$$\begin{aligned} bias^2(\boldsymbol{x}) &= (E_{y|\boldsymbol{x}}\{y\} - E_{LS}\{\hat{y}(\boldsymbol{x})\})^2 \\ &= (f(\boldsymbol{x}) - E_{LS}\{\boldsymbol{x}^T \boldsymbol{w}_{OLS}\})^2 \\ &= (f(\boldsymbol{x}) - E_{LS}\{\boldsymbol{x}^T (\arg\min_{\boldsymbol{w}\in\mathbb{R}^p} \sum_{i=1}^{N} (y_i - \boldsymbol{x}_i^T \boldsymbol{w})^2)\})^2 \end{aligned}$$

$$\begin{aligned} variance(\boldsymbol{x}) &= E_{LS}\Big\{(\hat{y}(\boldsymbol{x}) - E_{LS}\{\hat{y}(\boldsymbol{x})\})^2\Big\} \\ &= E_{LS}\Big\{(\boldsymbol{x}^T \boldsymbol{w}_{OLS} - E_{LS}\{\boldsymbol{x}^T \boldsymbol{w}_{OLS}\})^2\Big\} \\ &= E_{LS}\Big\{(\boldsymbol{x}^T (\arg\min_{\boldsymbol{w}\in\mathbb{R}^p} \sum_{i=1}^{N} (y_i - \boldsymbol{x}_i^T \boldsymbol{w})^2) - E_{LS}\{\boldsymbol{x}^T (\arg\min_{\boldsymbol{w}\in\mathbb{R}^p} \sum_{i=1}^{N} (y_i - \boldsymbol{x}_i^T \boldsymbol{w})^2)\})^2\Big\} \end{aligned}$$

- When $\lambda \neq 0$, we have :

$$\begin{aligned} bias^2(\boldsymbol{x}) &= (E_{y|\boldsymbol{x}}\{y\} - E_{LS}\{\hat{y}(\boldsymbol{x})\})^2 \\ &= (f(\boldsymbol{x}) - E_{LS}\{\frac{\boldsymbol{x}^T}{1+\lambda} \boldsymbol{w}_{OLS}\})^2 \\ &= (f(\boldsymbol{x}) - E_{LS}\{\frac{\boldsymbol{x}^T}{1+\lambda} (\arg\min_{\boldsymbol{w}\in\mathbb{R}^p} \sum_{i=1}^{N} (y_i - \boldsymbol{x}_i^T \boldsymbol{w})^2)\})^2 \end{aligned}$$

$$\begin{aligned} variance(\boldsymbol{x}) &= E_{LS}\Big\{(\hat{y}(\boldsymbol{x}) - E_{LS}\{\hat{y}(\boldsymbol{x})\})^2\Big\} \\ &= E_{LS}\Big\{(\frac{\boldsymbol{x}^T}{1+\lambda} \boldsymbol{w}_{OLS} - E_{LS}\{\frac{\boldsymbol{x}^T}{1+\lambda} \boldsymbol{w}_{OLS}\})^2\Big\} \\ &= E_{LS}\Big\{(\frac{\boldsymbol{x}^T}{1+\lambda} (\arg\min_{\boldsymbol{w}\in\mathbb{R}^p} \sum_{i=1}^{N} (y_i - \boldsymbol{x}_i^T \boldsymbol{w})^2) - E_{LS}\{\frac{\boldsymbol{x}^T}{1+\lambda} (\arg\min_{\boldsymbol{w}\in\mathbb{R}^p} \sum_{i=1}^{N} (y_i - \boldsymbol{x}_i^T \boldsymbol{w})^2)\})^2\Big\} \end{aligned}$$

2 Empirical analyses

In this part, we consider another regression problem $y = f(x) + \varepsilon$ where $\varepsilon \sim \mathcal{N}(0, \sigma^2)$ and $x \sim \mathcal{U}(0,2)$. We denote by $LS = \{(x_i, y_i) | i = 1, ..., N\}$ the learning sample of size N and by $\mathscr{A}$ a supervised learning algorithm.

(a) *Expressions*

As $\varepsilon \sim \mathcal{N}(0, \sigma^2)$, we know that $E(\varepsilon) = 0$ and $Var(\varepsilon) = \sigma^2$ so we can derive an expression for the expected prediction error of a regression fit $\hat{f}(x)$ at a given input point x_0, using squared-error loss :

$$Err(x_0) = E\{(y - \hat{f}(x_0))^2 | x = x_0\}$$

When averaging over all learning sets :

$$\begin{aligned} Err(x_0) &= E_{LS}\Big\{E\{(y - \hat{f}(x_0))^2 | x = x_0\}\Big\} \\ &= var_{y|x_0}\{y\} + bias^2(x_0) + var_{LS}\{\hat{y}(x_0)\} \end{aligned}$$

Such as we can identify the following terms :

- Residual error :

  $$var_{y|x_0}\{y\} = var_{y|x_0}\{f(x_0) + \varepsilon\} = var_{y|x_0}\{f(x_0)\} + var_{y|x_0}\{\varepsilon\} = \sigma^2$$

  Because $f(x_0)$ and ε are independant and $f(x_0)$ is a constant when x_0 is fixed.
  $\rightarrow$ This is the variance of the target around its true mean $f(x_0)$, which cannot be avoided no matter how well we estimate $f(x_0)$, unless $\sigma^2 = 0$.

- Squared Bias :

  $$\begin{aligned} bias^2(x_0) &= (E_{y|x0}\{y\} - E_{LS}\{\hat{y}(x_0)\})^2 = (E_{y|x0}\{f(x_0) + \varepsilon\} - E_{LS}\{\hat{f}(x_0)\})^2 \\ &= (f(x_0) - E_{LS}\{\hat{f}(x_0)\})^2 \end{aligned}$$

  Because the mean of ε is 0.
  $\rightarrow$ This is the amount by which the average of our estimate differs from the true mean.

- Variance :

  $$var_{LS}\{\hat{y}(x_0)\} = E\{(\hat{f}(x_0) - E_{LS}\{\hat{f}(x_0))^2\}\}$$

  $\rightarrow$ This is the expected squared deviation of $f(x_0)$ around its mean.

The expected prediction error can thus be written such as :

$$Err(x_0) = \text{Residual error} + \text{Squared Bias} + \text{Variance}$$

(b) *Protocol*

From the expressions given in the previous question, we can notice that in order to compute an estimate of the squared bias and the variance at a given point x_0, we firstly need estimations of the Bayes model $f(x_0)$ and the average model $E_{LS}\{\hat{f}(x_0)\}$.

We can use the following experimental protocol for all x_0 in the LS :

1. Create a large dataset of size N by generating N input values $x \sim \mathscr{U}(0,2)$ and using the expression $y = f(x) + \varepsilon$ to compute the associated output values. The dataset is thus composed of N samples (x_i, y_i).
2. Select all the samples (x_i, y_i) for which $x_i = x_0$.
3. Compute the mean and the variance of this subset. The mean $E_{y|x_0}\{y\}$ corresponds to the Bayes model and the variance $V_{y|x_0}\{y\}$ corresponds to the variance of $y(x_0)$ around its true mean $f(x_0)$, which is the residual error (i).
4. Split the initial dataset of size N into k subsets.
5. For each of these subsets :
   - Train the algorithm such as we get samples $(x_i, \hat{y}(x_i))$.
   - Select the samples $(x_0, \hat{y}(x_0))$.
6. Compute the mean and the variance (iii) of $\hat{y}(x_0)$ over the k subsets, which are respectively $E_{LS}\{\hat{y}(x_0)\}$ and $V_{LS}\{\hat{y}(x_0)\}$.
7. Compute the squared difference of the Bayes model and the mean computed at step *6*, giving the squared bias (ii).
8. Computed the expected prediciton error (iv) by summing the residual error (step 3), the squared bias (step 7) and the variance (step 6).

Let us now assume that :

$$f(x) = -x^3 + 3x^2 - 2x + 1 \text{ and } \sigma^2 = 0.1$$

And let us consider the models :

$$\hat{y}_m(x) = \textstyle\sum_{i=0}^{m} a_i x^i \text{ of increasing degrees } 0 \leq m \leq 5.$$

(c) *Bayes model and residual error*

Script Q2_c.py

From now, we assume that $f(x) = -x^3 + 3x^2 - 2x + 1$ and $\sigma^2 = 0.1$. To answer this question, we have implemented the three first steps of the protocol : we generate a dataset of size $N = 30000$ containing the samples (x_i, y_i). We can see its shape on the figure below :

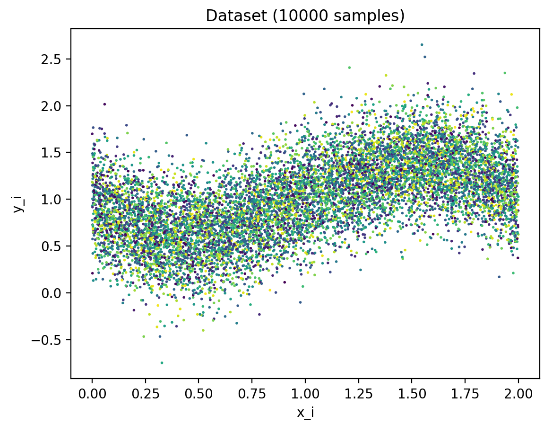


Figure 1

Then, we select the samples for which $x_i = x_0$ and we compute the mean and the variance of the corresponding y_0. This is done in a loop for all x_0 (from 0 to 2 with a step of 0.05). By doing so, we obtain respectively estimates of the Bayes model and the residual error, that we compare to the analytical values :

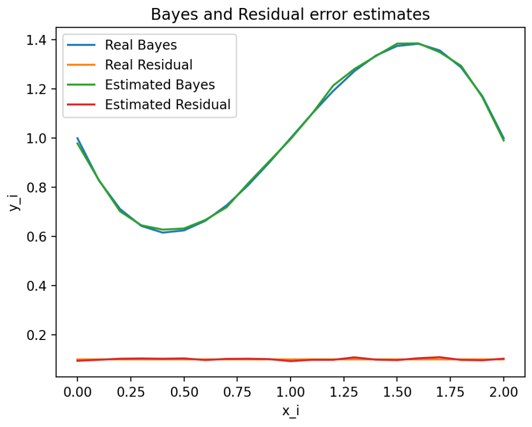


Figure 2

We can see that, when N is large enough, the estimations are really close to the real values. In fact, as the Bayes model is $E_{y|x_0}\{y\}$, in practice we get :

$$mean\{y(x_0)\} = mean\{f(x_0) + \varepsilon\} = mean\{f(x_0)\} + mean\{\varepsilon\}$$

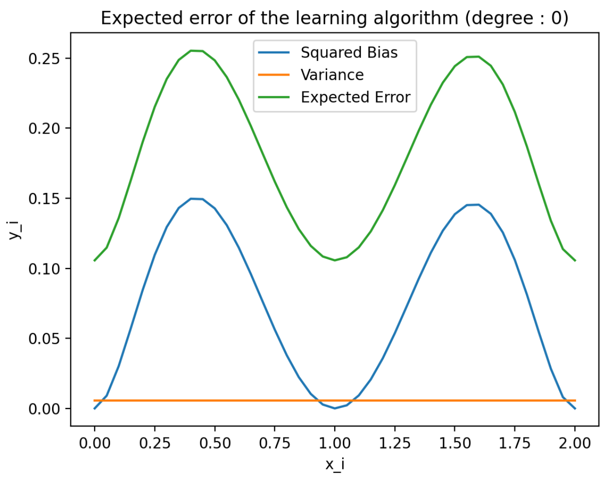
As $\varepsilon \sim \mathcal{N}(0, \sigma^2)$, its mean get ideed closer to 0 as the size of the dataset increases, leaving the estimated mean and the estimated variance of $f(x_0)$ and really close to their theoretical values.

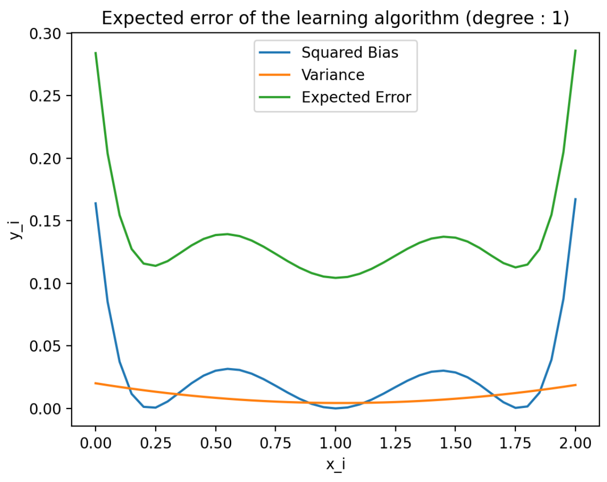
(d) *Estimated squared bias, variance and error*

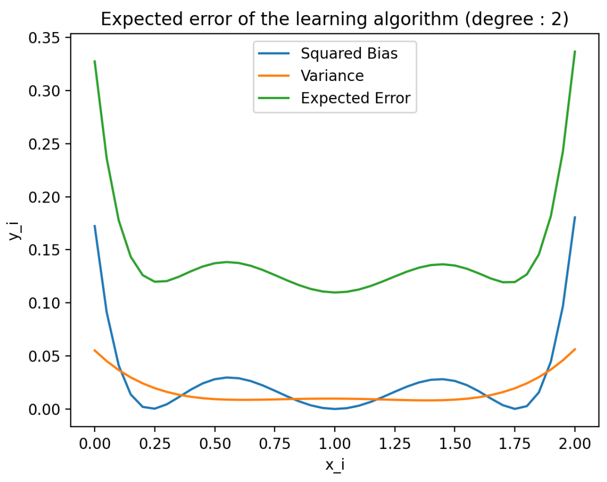
Script Q2_d_e.py

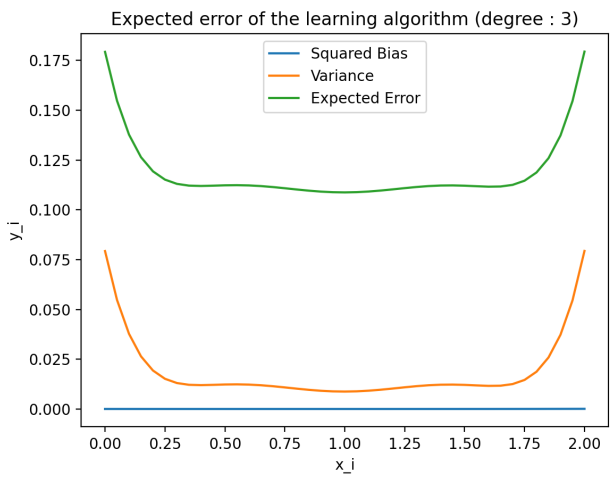
In this section, we consider as candidate models $\hat{y}_m(x) = \sum_{i=0}^{m} a_i x^i$ of increasing degrees $0 \leq m \leq 5$, fitted by ordinary least-square.

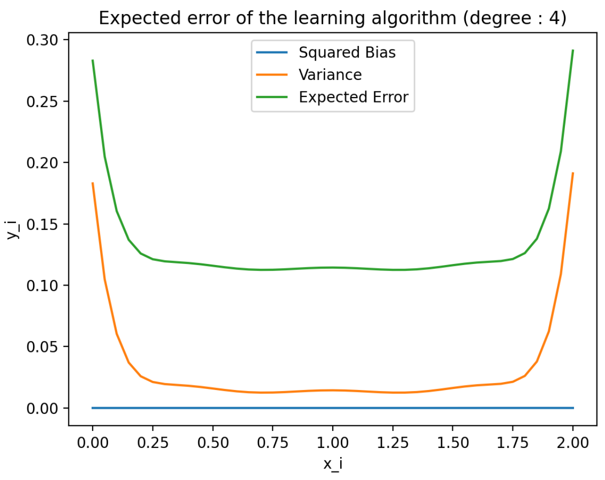
In order to estimate the squared bias, the variance and the expected error of $\hat{y}$, we have implemented the steps 4 to 8 of the protocol. So, once the dataset set of size $N = 30000$ is generated, we select the first 30th samples (x_i, y_i), which represent the first learning sample. Then, a regression model of the current considered degree is fitted to the data by ordinary least-square, with the *LinearRegression()* function. Then, the model is tested on the $x0$ values $\in [0, 2]$ with a step of 0.05 such as we get the corresponding estimates $\hat{y}_m(x_0)$ of this model trained on this LS. This model fitting is repeated for every *LS*, such as we can finally compute the mean of the $\hat{y}_m(x_0)$ over the LS and for all x_0 values. Then, the squared bias, the variance and the residual error are computed, and represented on the following figures :











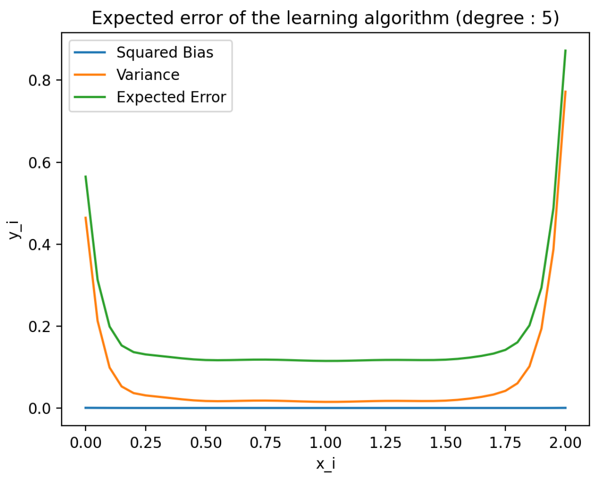


Figure 3

We can notice that, for a degree = 0, the squared bias is very high, meaning that the model is very bad at fitting the data since it's a straight constant line but the variance is very low, because the model training for each LS leads to very similar models (the model has only 1 parameter). A low variance and high bias leads to underfitting : the model does not fit the data well enough.

For a degree = 1, the variance is a bit larger than for the degree = 0, but it's still small because the model is a linear regression, and thus has only few parameters. Thus, it has a small variance. However, since the true function (bayes model) is non-linear, it still has a high squared bias, because the model can't fit well the data. .

For a degree = 2, we can notice that the quantities are very similar to the ones for a degree = 1. In fact, when we think about the symmetrical shape of the dataset on *Figure 1*, we can guess that the best polynomial model fit of degree 2 is still a quasi-1 degree function. This behavior is illustrated on the following figure :

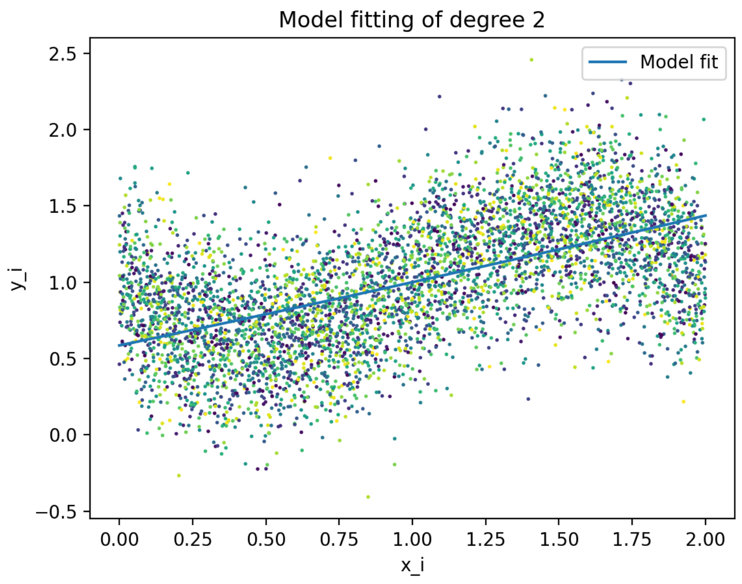


Figure 4

For the higher degrees, we can see that the square bias is low and the variance is increasing with the degree. To better discuss the impact of model complexity on these quantities, the following figures show the quantities w.r.t the degree for particular values of x :

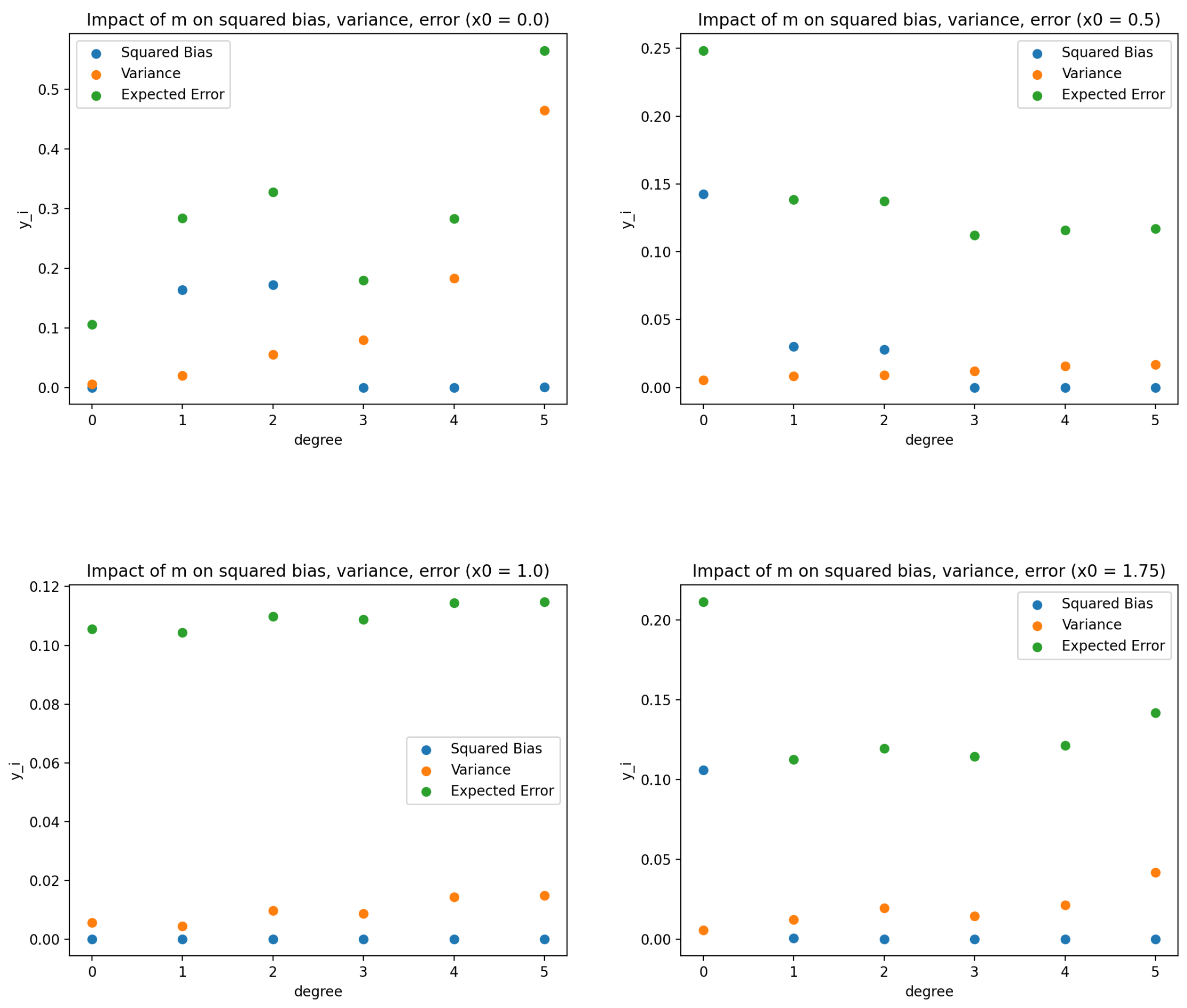


Figure 5

At first, we can notice that on the four graphs, when the model complexity increases, the variance of $\hat{y}$ increases while the squared bias decreases. We can understand that, on one hand, the variance is an increasing function of the complexity because it represents the variability of the model with respect to the learning sample randomness. Thus, when the model complexity is higher, it has more parameters, and thus a larger variability. On the other hand, the bias is a decreasing function of the complexity because a more complex model can better learns the detail and noise in the training data (but when the complexity is too high, this usually negatively impacts the performance of the model on new data = overfitting).

Then, we can discuss the impact of model complexity particular to the given x_0 values :

For $x_0 = 0$, the complexity leading to the minimal expected error is 0. This means that the variance is very low (which is not surprising since the model has only 1 parameter) and the squared bias is very low too! Actually, this is simply because the model luckily fits the data efficiently for this input, but as the model is a constant line, as soon as the true value of the input $f(x_0)$ digress from this constant, the model don't fit the data no more and the squared bias becomes very high, as can be seen for $x_0 = 0.5$ and $x_0 = 1.75$.

For $x_0 = 1$, the squared bias is very low for all complexities, which means that the 6 models fit well the data at this specific input value (they all predict precisely $f(1)$).

Finally, when we look at $x_0 = 1$ and $x_0 = 1.75$, the model of complexity 1 leads to the lowest expected error, but for $x_0 = 0.5$, it's the model of complexity 3 which is optimal. To be able to know which model is the optimal over the whole input space, we need to calculate the means of these quantities over it. That's what we'll do in the next section.

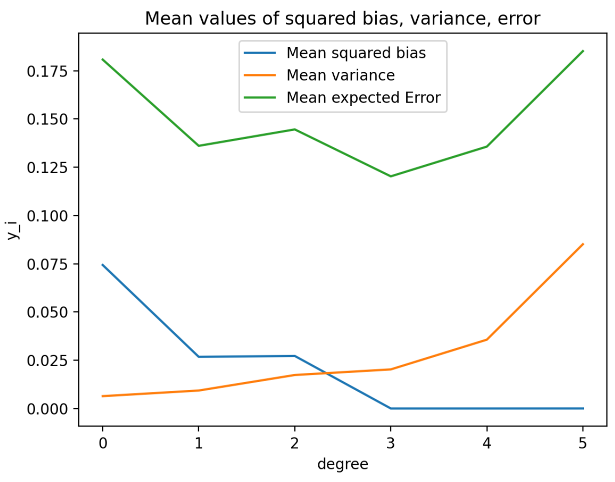
(e) *Estimated mean values of the previous quantities over the input space*

Script Q2_d_e.py

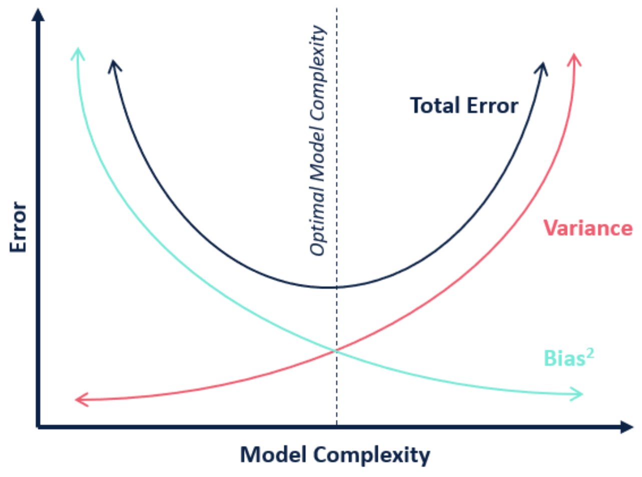
In order to estimate the mean values of the previous quantities over the input space, we can adapt the protocol of *question (b)* : the steps 2 to 8 are repeated for every x0 in the input space and for each degree in $[0,5]$ (*in practice, we repeat these steps in a for loop over the values of* x_0 (*from* 0 *to* 2 *with a step of* 0.05) *which is inside another for loop over the m degrees*). That leads us to the following adapted protocol (steps in grey remain unchanged) :

1. Create a large dataset composed of N samples (x_i, y_i).
2. For each degree in $[0,5]$:
    - Split the initial dataset of size N into k subsets.
    - For each of these subsets :
        - Train the algorithm such as we get samples $(x_i, \hat{y}_m(x_i))$.
        - For each value of $x_0 \in [0,2]$ with a step of 0.05 :
            - Test the model such as we get the samples $(x_0, \hat{y}_m(x_0))$
    - Compute the squared bias, the variance and the expected error of $\hat{y}(x_0)$ for each x_0 values over the k subsets.
    - Compute the means of these values over the whole input space.

By applying this protocol, we finally get the means of the squared bias, the variance and the expected error over the input space for each considered degree. We thus have the following results, presented together with the general behavior theoretical graph[1] :



Experimental



Theoretical

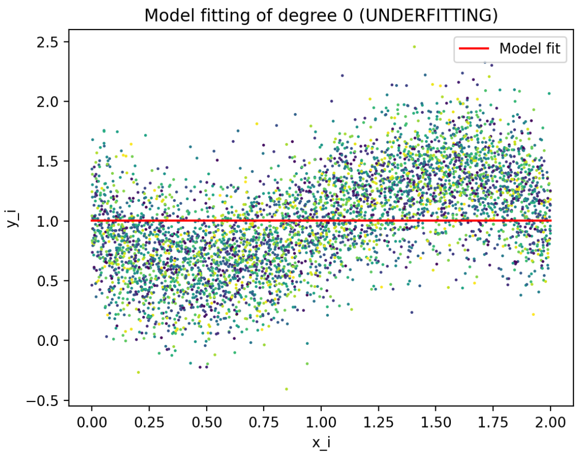
Figure 6

[1]https://community.alteryx.com/t5/Data-Science/Bias-Versus-Variance/ba-p/351862

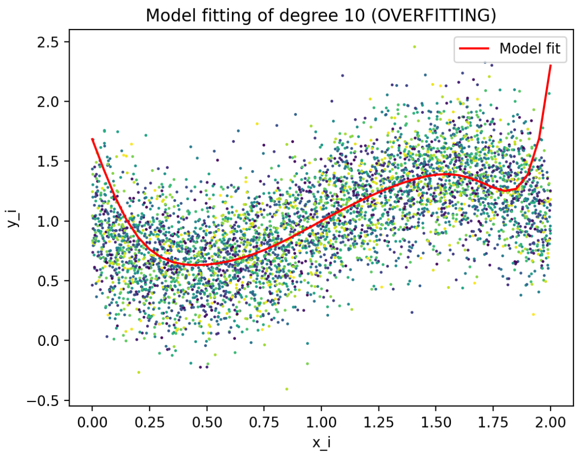
On this figure, we can observe that our results match the theory : the variance is an increasing function of the complexity while the bias is a decreasing function of the complexity.

As the expected total error is the sum of these 2 elements (and the residual error which is a constant), its minimum corresponds to the optimal tradeoff between both (the balance between the variance and the bias that minimizes overall error). This optimal tradeoff is met with a certain model complexity, which is thus the optimal model complexity. When the complexity becomes too high, we can see that the bias is very low but the variance is very high, which is a consequence of overfitting : the model gets trained with too much data and starts learning from the noise in the data set. Inversly, when the complexity is too low, we can see that the variance is very low but the bias is very high, which is a consequence of underfitting : the model don't capture the underlying trend of the data.

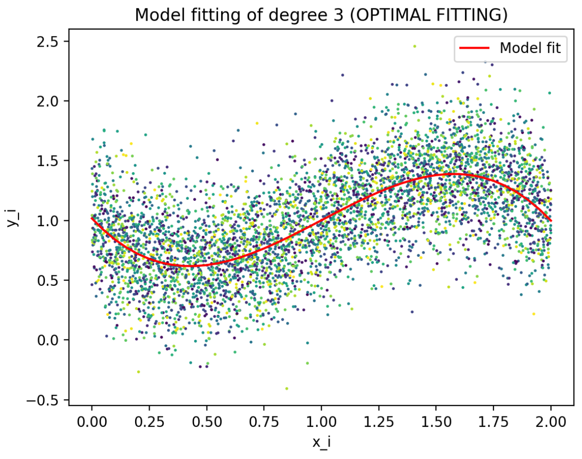
Consequently, on our experimental results on *Figure 5*, we can see that the minimum expected error of the algorithm is met when the degree is equal to 3. Thus, in our case, a polynomial regression of degree 3 is the most efficient. To illustrate this, we have plotted 3 model fitting when the degree is 0, 3 and 10. We can easily see respectively the underfitting, the optimal fitting and the overfitting :



degree 0 : underfitting



degree 10 : overfitting



degree 3 : optimal

Figure 7

(f) *Ridge regression*

Script Q2_f.py

For this section, we use the same protocol as the one described in the previous question. But, instead of using the ordinary least-square method (implemented in Scikit-learn by the function LinearRegression), we are using the ridge regression (implemented in Scikit-learn by the function Ridge) to fit the models and we only consider a model complexity of the $5th$ degree.

The ridge regression uses a penalty term which reduces the coefficient values towards zero to allow the less contributive variables to have a coefficient close to zero or equal zero. The penalty term is tuned by the parameter λ. When $\lambda = 0$, ridge regression equals least squares regression and when $\lambda \to \infty$, all coefficients are shrunk to zero. We have observed empirically the effect of this regularisation level $\lambda \in [0,2]$ on the squared bias, the variance, and the error. On the following figures, we firstly present the quantities of interest of the algorithm with $\lambda = 0$, 1 and 2 w.r.t the values of $x_0 \in [0,2]$ (*Figure 8*) and secondly, the mean values of these quantities over the input space w.r.t λ (*Figure 9*) :

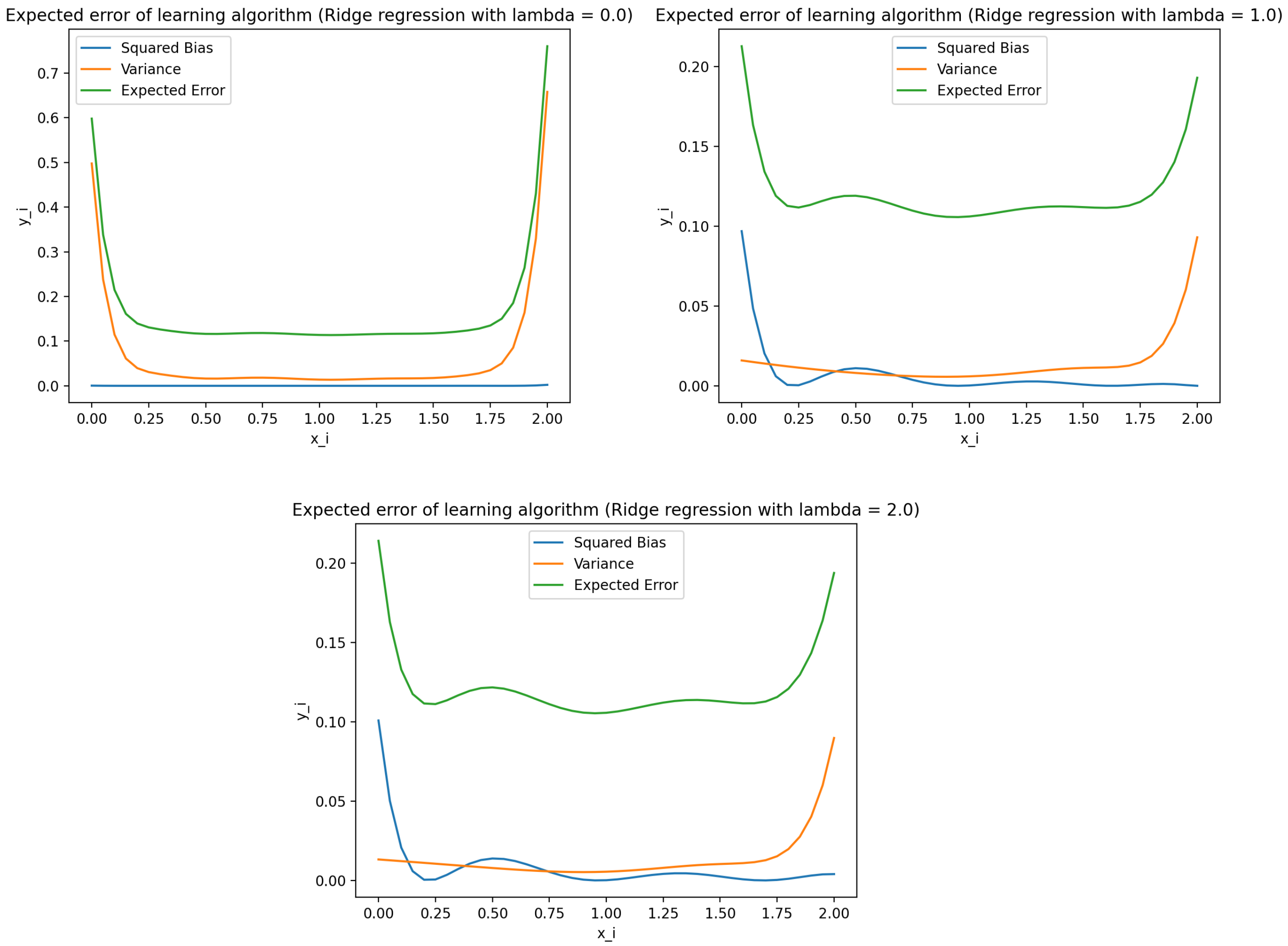


Figure 8

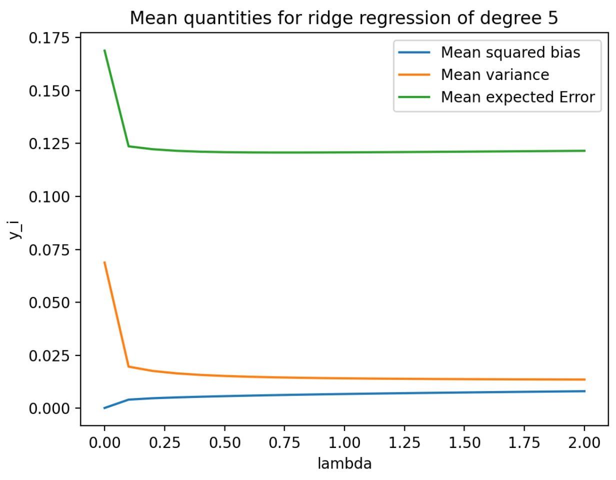


Figure 9

Firstly, we can notice that on the *Figure 8*, for $\lambda = 0$ (and degree = 5), we indeed have the same results as the ones we obtained with the ordinary least-square (with degree = 5) on the *Figure 3*. Secondly, on the *Figure 8*, when $\lambda = 1$ or 2, the quantities don't seem to change a lot.

On the *Figure 9*, we can also see that when $\lambda = 0$, the expected error is $\simeq$ to the expected error for a linear regression of the same degree, but as soon as λ increases, the value of the expected error drops and then remains quite constant (increases very slightly).

Consequently, this means that when the model complexity is high, the ridge regression is more efficient than the ordinary least-square. In fact, the latter doesn't differentiate "important" from "less-important" predictors in a model, so it includes all of them. Thus, as we could see in question *2.(d)*, the bias of the estimates is very low but the variance is very high, such as the expected error becomes very high too, leading to overfitting. On the opposite, ridge regression adds just enough bias to make the estimates reasonably reliable approximations to true $f(x)$ values.