University of Liège
School of Engineering

Project 2 - Bias and variance analysis

ELEN0062: Introduction to machine learning

Julien Gustin, Joachim Houyon

November 21, 2021

1 Analytical derivations

1.1 Bayes model and residual error in classification

(a) Lets defines the **Bayes model** as $h_B(x_1, x_2) = \text{argmax}_c\, P(y = c|\boldsymbol{x})$ where $y = \{0, 1\}$ and for which $\boldsymbol{x} = [x_1, x_2]^T$, $\boldsymbol{\mu}^y = [\mu_1^y, \mu_2^y]^T$, $\boldsymbol{\mu}^{y=0} = [1.5, 1.5]^T$ and $\boldsymbol{\mu}^{y=1} = [-1.5, -1.5]^T$.

$$\boldsymbol{x}|y \sim \mathcal{N}(\boldsymbol{\mu^y}, \sum) \tag{1}$$

$$\sum = \begin{pmatrix} \sigma^2 & 0 \\ 0 & \sigma^2 \end{pmatrix} \tag{2}$$

Note that the covariance matrix is the same for *y={0, 1}* and that $Cov(x_1, x_2) = 0$ for the sake of simplicity let replace $y = c$ by y

$$\begin{aligned} h_B(x_1, x_2) &= \underset{y}{\text{argmax}}\ p(y = c|x_1, x_2) \\ &= \underset{y}{\text{argmax}}\ \frac{p(x_1, x_2|y) \times P(y)}{P(x_1, x_2)} \quad \textit{P(}x_1, x_2\textit{) do not depend of y} \\ &= \underset{y}{\text{argmax}}\ p(x_1, x_2|y) \times P(y) \quad \textit{We have 3} \times \textit{ more y=0 than y=1} \\ &= \underset{y}{\text{argmax}}\ p(x_1, x_2|y) \times (y \times 0.25 + (1 - y) \times 0.75) \end{aligned} \tag{3}$$

From **(1)** and **(2)** we know that $p(x1, x2|y) = \frac{1}{2\pi\sigma^2\sqrt{(1-\rho^2)}} exp(-\frac{z}{2(1-\rho)^2})$ where $z = \frac{(x_1^y-\mu_1^y)^2 - 2\rho(x_1^y-\mu_1^y)(x_2^y-\mu_2^y)+(x_2^y-\mu_2^y)^2}{\sigma^2}$ and $\rho = \frac{Cov(x_1^y, x_2^y)}{\sigma^2} = 0$. This give us

$$p(x_1, x_2|y) = \frac{1}{2\pi\sigma^2} \times exp\left(-\frac{1}{2} \times \frac{(x_1 - \mu_1)^2 + (x_2 - \mu_2)^2}{\sigma^2}\right) \tag{4}$$

When substituting $p(x1, x2|y)$ of **(3)** by **(4)** and removing *useless* constant we have

$$h_B(x_1, x_2) = \underset{y}{\text{argmax}}\ exp\left(-\frac{1}{2} \times \frac{(x_1 - \mu_1)^2 + (x_2 - \mu_2)^2}{\sigma^2}\right) \times (y \times 0.25 + (1-y) \times 0.75)$$

that can also be re-write as

$$h_B(x1, x_2) = \begin{cases} 1, & \text{if } p(y = 1|x_1, x_2) \geq p(y = 0|x_1, x_2). \\ 0, & \text{otherwise.} \end{cases} \tag{5}$$

$$p(x_1, x_2|y = 1) \times P(y = 1) = exp\left(-\frac{1}{2} \times \frac{(x_1 - \mu_1^{y=1})^2 + (x_2 - \mu_2^{y=1})^2}{\sigma^2}\right) \times 0.25$$

$$p(x_1, x_2|y = 0) \times P(y = 0) = exp\left(-\frac{1}{2} \times \frac{(x_1 - \mu_1^{y=0})^2 + (x_2 - \mu_2^{y=0})^2}{\sigma^2}\right) \times 0.75$$

Lets focus on the case when $h_B(x_1, x_2)$ predict **y=1** i.e. the positive class0:

$$p(y = 1|x_1, x_2) \geq p(y = 0|x_1, x_2)$$

let $z_1 = (x_1 - \mu_1^{y=1})^2 + (x_2 - \mu_2^{y=1})^2$ and $z_0 = (x_1 - \mu_1^{y=0})^2 + (x_2 - \mu_2^{y=0})^2$.
This give us:

$$\begin{aligned}
&exp\left(-\frac{1}{2} \times \frac{z_1}{\sigma^2}\right) \times 0.25 \geq exp\left(-\frac{1}{2} \times \frac{z_0}{\sigma^2}\right) \times 0.75 \\
\Leftrightarrow &-\frac{1}{2} \times \frac{z_1}{\sigma^2} + ln(0.25) \geq -\frac{1}{2} \times \frac{z_0}{\sigma^2} + ln(0.75) \quad \textit{Apply ln on both side} \\
\Leftrightarrow &\frac{z_1}{\sigma^2} \leq \frac{z_0}{\sigma^2} - 2 \times (ln(0.75) - ln(0.25)) \quad \textit{Multiply by -2 on both side} \\
\Leftrightarrow &z_1 \leq z_0 - 2 \times \sigma^2 \times ln\left(\frac{0.75}{0.25}\right) \quad \textit{Multiply by } \sigma^2 \textit{ } (\geq 0) \textit{ on both side} \\
&\textit{By substituting } z_i \textit{ and } \mu^y \textit{ this give us} \\
\Leftrightarrow &(x_1 + 1.5)^2 + (x_2 + 1.5)^2 \leq (x_1 - 1.5)^2 + (x_2 - 1.5)^2 - 2 \times \sigma^2 \times ln\left(\frac{0.75}{0.25}\right) \\
\Leftrightarrow &x_1^2 + 2 \times x_1 \times 1.5 + 1.5^2 + x_2^2 + 2 \times x_2 \times 1.5 + 1.5^2 \leq \\
&x_1^2 - 2 \times x_1 \times 1.5 + 1.5^2 + x_2^2 - 2 \times x_2 \times 1.5 + 1.5^2 - 2 \times \sigma^2 \times ln\left(\frac{0.75}{0.25}\right) \\
\Leftrightarrow &x_1 \times 1.5 + x_2 \times 1.5 \leq -x_1 \times 1.5 - x_2 \times 1.5 - \sigma^2 \times ln\left(\frac{0.75}{0.25}\right) \\
\Leftrightarrow &x_1 \times (1.5 + 1.5) + x_2 \times (1.5 + 1.5) \leq -\sigma^2 \times ln\left(\frac{0.75}{0.25}\right) \quad \textit{isolate } x_1 \textit{ and } x_2 \\
\Leftrightarrow &x_1 \times 3 + x_2 \times 3 \leq -\sigma^2 \times ln\left(\frac{0.75}{0.25}\right) \\
\Leftrightarrow &x_1 + x_2 \leq \frac{-\sigma^2}{3} \times ln\left(\frac{0.25}{0.75}\right) \quad \textit{Multiply by 3 on both side}
\end{aligned}$$

This give us:

$$h_B(x_1, x_2) = \begin{cases} 1, & \text{if } x_1 + x_2 \leq \frac{-\sigma^2}{3} \times ln\left(\frac{0.75}{0.25}\right). \\ 0, & \text{otherwise.} \end{cases} \tag{6}$$

(b) (See *src/1b_d.ipynb for the script*)

As we can see in the right hand side we have the coefficient $ln\left(\frac{0.75}{0.25}\right)$, meaning that the ratio of the negative and positive class has an impact to the solution. The following plots will give us more intuition of what happen when the ratio change.

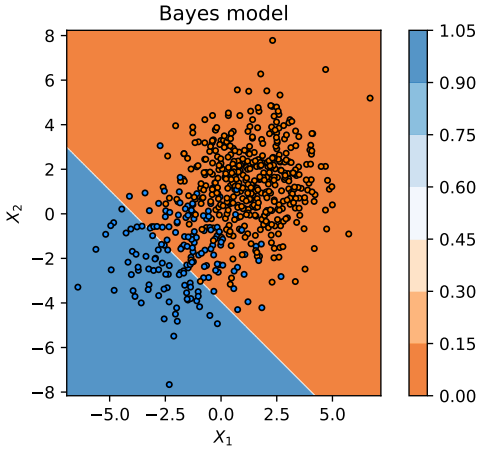


Figure 1. $ratio = \frac{0.99}{0.01}$

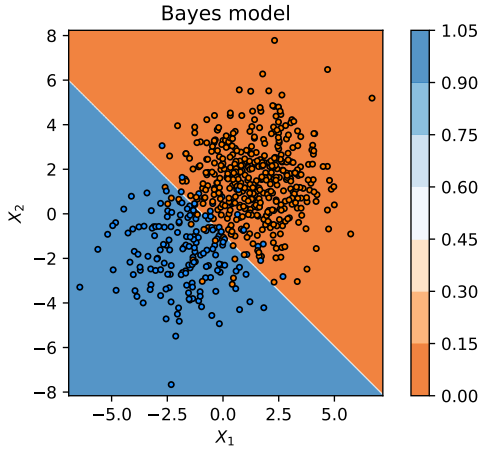


Figure 2. $ratio = \frac{0.75}{0.25}$

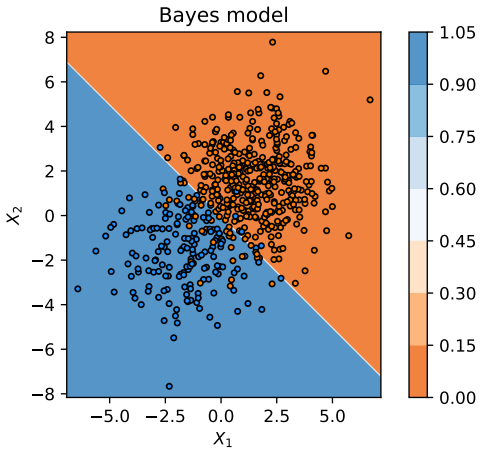


Figure 3. $ratio = \frac{0.5}{0.5}$

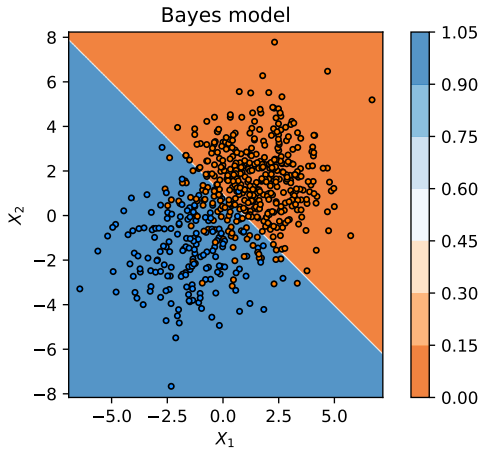


Figure 4. $ratio = \frac{0.25}{0.75}$

Figure 5. Boundary given different ratio between the negative and positive class (orange is the negative class)

As we can see the slope of the lines separating the two classes are identical, however the offset change in the direction of the most abundant class. This is expected because the model will privilige the class that have the highest ratio in order to minimize the zero-one error loss.

(c) The residual error is defined as $E_{x_1,x_2,y}\{\mathbf{1}(y \neq h_B(x_1,x_2))\}$

$$\begin{aligned}
E_{x_1,x_2,y}\{\mathbf{1}(y \neq h_B(x_1,x_2))\} &= P(y \neq h_B(x_1,x_2)) \\
&= \sum_y P(y \neq h_B(x_1,x_2)|y) \times P(y) \\
&= P\left(x_1 + x_2 \leq \frac{-\sigma^2}{3} \times ln\left(\frac{0.75}{0.25}\right)|y=0\right) \times P(y=0) \\
&\quad + P\left(x_1 + x_2 > \frac{-\sigma^2}{3} \times ln\left(\frac{0.75}{0.25}\right)|y=1\right) \times P(y=1) \\
&= P\left(x_1 + x_2 \leq \frac{-\sigma^2}{3} \times ln\left(\frac{0.75}{0.25}\right)|y=0\right) \times 0.75 \\
&\quad + P\left(x_1 + x_2 > \frac{-\sigma^2}{3} \times ln\left(\frac{0.75}{0.25}\right)|y=1\right) \times 0.25
\end{aligned}$$

We know that $x_1 \perp x_2$ as $Cov(x_1,x_2) = 0$ therefore:

$$x_1|y \sim \mathcal{N}(\mu_1^y, \sigma^2) \;\; and \;\; x_2|y \sim \mathcal{N}(\mu_2^y, \sigma^2)$$

let $t = x_1 + x_2$ such that

$$t|y \sim \mathcal{N}(\mu_1^y + \mu_2^y, 2\sigma^2)$$

$$\begin{aligned}
E_{x_1,x_2,y}\{\mathbf{1}(y \neq h_B(x_1,x_2))\} &= P\left(t \leq \frac{-\sigma^2}{3} \times ln\left(\frac{0.75}{0.25}\right)|y=0\right) \times 0.75 \\
&\quad + P\left(t > \frac{-\sigma^2}{3} \times ln\left(\frac{0.75}{0.25}\right)|y=1\right) \times 0.25
\end{aligned}$$

Where

$$P\left(t \leq \frac{-\sigma^2}{3} \times ln\left(\frac{0.75}{0.25}\right)|y=0\right) = \frac{1}{\sqrt{2\sigma^2}\sqrt{2\pi}} \int_{-\infty}^{\frac{-\sigma^2}{3} \times ln\left(\frac{0.75}{0.25}\right)} exp\left(\frac{-1}{2}\left(\frac{t-3}{\sqrt{2\sigma^2}}\right)^2\right) dt$$

and

$$P\left(t > \frac{-\sigma^2}{3} \times ln\left(\frac{0.75}{0.25}\right)|y=1\right) = \frac{1}{\sqrt{2\sigma^2}\sqrt{2\pi}} \int_{\frac{-\sigma^2}{3} \times ln\left(\frac{0.75}{0.25}\right)}^{+\infty} exp\left(\frac{-1}{2}\left(\frac{t+3}{\sqrt{2\sigma^2}}\right)^2\right) dt$$

(d) (See *src/1b_d.ipynb for the script*)

Using *scipy.integrate.quad* we have

```
from scipy.integrate import quad.
import numpy as np

sigma_2 = 1.6**2

lower_bound_positive = (-sigma_2)/3 * np.log(0.75/0.25)
upper_bound_positive = np.inf

lower_bound_negative = -np.inf
upper_bound_negative = (-sigma_2)/3 * np.log(0.75/0.25)

mu_positive = -3
mu_negative = 3

f_positive = lambda t: np.exp(-1/2 * ((t -
    mu_positive)/(np.sqrt(2*sigma_2)))**2)
f_negative = lambda t: np.exp(-1/2 * ((t -
    mu_negative)/(np.sqrt(2*sigma_2)))**2)

p_y_0 = 1/(np.sqrt(2*sigma_2) * np.sqrt(2*np.pi)) *
    quad(f_negative, lower_bound_negative, upper_bound_negative)[0]
p_y_1 = 1/(np.sqrt(2*sigma_2) * np.sqrt(2*np.pi)) *
    quad(f_positive, lower_bound_positive, upper_bound_positive)[0]

print(p_y_0 * 0.75 + p_y_1 * 0.25) # 0.0759
```

Meaning:

$$P\left(t \leq \frac{-\sigma^2}{3} \times log\left(\frac{0.75}{0.25}\right) | y = 0\right) \times 0.75 + P\left(t > \frac{-\sigma^2}{3} \times ln\left(\frac{0.75}{0.25}\right) | y = 1\right) \times 0.25 \approx 7.59\%$$

When computing it empirically with a sample size of 100000 samples we have a generalization error of 7.59%. Which coincide to the one computed above !

We can conclude that the error of the bayes model is 7.59%

1.2 Bias and variance in regression

(a) Recall that the Bayes model $h_B(x) = E_{y|x}[y]$ and noise(x) $= E_{y|x}[(y - E_{y|x}[y])^2]$

We get that $h_B(x)$ is

$$\begin{aligned} h_B(x) &= E_{y|x}[ax + \epsilon] \\ &= E_{y|x}[ax] + E_{y|x}[\epsilon] \\ &= ax \end{aligned} \tag{7}$$

Now we can compute the residual error:

$$\begin{aligned} noise(x) &= E_{y|x}[(ax + \epsilon - ax)^2] \\ &= E_{y|x}[\epsilon^2] \\ &= Var_{y|x}[\epsilon] + E_{y|x}[\epsilon]^2 \\ &= \sigma^2 \end{aligned} \tag{8}$$

(b) Before computing the mean squared bias and variance of our learning algorithm, let's prove that the value of μ is always equal to the mean of the given learning set $\bar{y}_{LS}$. Indeed, the loss function is the square error and we want to minimize the following:

$$\begin{aligned} &\operatorname*{argmin}_{\mu} \sum_{i=1}^{N} (y_i - \mu)^2 \\ &= \operatorname*{argmin}_{\mu} \sum_{i=1}^{N} y_i^2 + \mu^2 - 2y_i\mu \\ &\Leftrightarrow \sum_{i=1}^{N} 2\mu - 2y_i = 0 \\ &\Leftrightarrow 2N\mu - 2\sum_{i=1}^{N} y_i = 0 \\ &\Leftrightarrow \mu = \frac{1}{N}\sum_{i=1}^{N} y_i \\ &\Leftrightarrow \mu = \bar{y}_{LS} \end{aligned} \tag{9}$$

Now, we can compute the bias:

$$\begin{aligned} bias^2(x) &= (h_B(x) - E_{LS}[\hat{f}_{LS}(x)])^2 \\ &= (h_B(x) - \frac{1}{N}\sum_{i=1}^{N} E_{y|x}[\hat{f}(x_i)])^2 \\ &= (ax - \bar{y}_{LS})^2 \end{aligned} \tag{10}$$

Since the input x is drawn uniformly in $[0, 1]$, we know if N goes to $+\infty$ that

$$\begin{aligned} \bar{y}_{LS} &= lim_{N\to+\infty} \frac{1}{N}\sum_{i=1}^{N} y_i \\ &= E_y[y] \\ &= aE_x[x] \\ &= a\frac{(1-0)}{2} \\ &= \frac{a}{2} \end{aligned} \tag{11}$$

Therefore, the minimum squared bias of our best estimator given this model complexity is

$$\begin{aligned} bias^2(x) &= (ax - \frac{a}{2})^2 \\ &= a^2(x - \frac{1}{2})^2 \end{aligned} \tag{12}$$

Averaged over all x:

$$\begin{aligned} E_x[bias^2(x)] &= a^2 Var_x[x] \\ &= \frac{a^2}{12} \end{aligned} \tag{13}$$

Regarding the variance of the learning sample, we already know that the variance of a learning sample with no inputs is proportional to the variance of the distribution. Since $\hat{f}(x)$ does not depend of the input x, we are in a similar case. Hence, we get that

$$\begin{aligned} variance(x) &= \frac{1}{N} noise(x) \\ &= \frac{\sigma^2}{N} \end{aligned} \tag{14}$$

(c) (a) **Bias**: We get that the squared bias evolves quadratically as a function of a when the sample size goes to $+\infty$. We can also notice that estimating our model with a constant is not the right choice, as we get a non null bias for our best possible model given this complexity.

(b) **Noise**: The noise is equal to the variance of the noise variable ϵ. It means that, even if we get to find the best possible model ($h_B(x)$) for our regression problem, the minimum error that we could get is equal to σ^2

(c) **Variance**: The expected variance between is a function of the sampling size N and the variance of noise in the data σ^2. Intuitively, this is logical because we could assume two learning samples with $\mu_1 = \bar{y}_{LS1}$ and $\mu_2 = \bar{y}_{LS2}$ respectively. The expected variance between these two samples would just be the squared difference of the means between the two samples $(\bar{y}_{LS1} - \bar{y}_{LS2})^2$. This quantity is supposed to be big if there is a lot of variance in the data (controlled by the parameter σ). In the other hand, The variability between two samples should be low if the dataset is big enough (controlled by the parameter N).

2 Empirical analysis

(a) Lets define some notation,

(x, y) are input output pair and x_0 is the point for which we want to estimate the following:

$$\begin{aligned} \textit{noise / residual error: } & E_{y|x}\left[(y - E_{y|x}[y])^2\right] \\ \textit{bias}^2\textit{: } & (E_{y|x}[y] - E_{LS}[\hat{y}(x)])^2 \\ \textit{variance: } & E_{LS}[(\hat{y}(x) - E_{LS}[\hat{y}(x)])^2] \end{aligned}$$

- $E_{LS}[.]$ is approximated by averaging over multiple (n) learning samples
- $E_{y|x}[.]$ is approximated by averaging over a learning sample

Here is the protocol for a chosen point x_0

(i) Generate a large number of samples of size N call LS

(ii) Then estimate $E_{y|x_0}[.]$ by taking the average of all y_{x0} generated by the data generator given x_0.

(iii) We do the same but instead taking the unbiased variance of y_{x0} over the LS, this give us the *noise*.

(iv) Split the LS into n sub LS_i of size $\frac{N}{n} = m$, $1 \leq i \leq n$

(v) For each LS_i use the same kind of regression model $m_i(x) \rightarrow \hat{y}_x^i$, train them on their respective LS_i

(vi) Then for each m_i, $1 \leq i \leq n$ generate $\hat{y}_{x0}^i$ such that $\hat{y}_{x_0}^i = m_i(x_0)$. $Var[\hat{y}_{x0}]$ give us the *variance* over all LS and the squared difference between the mean of $\hat{y}_{x0}$ and the mean of y_{x0} is the $bias^2$.

(b) We can use the protocol we defined in (a) but for each x_i:

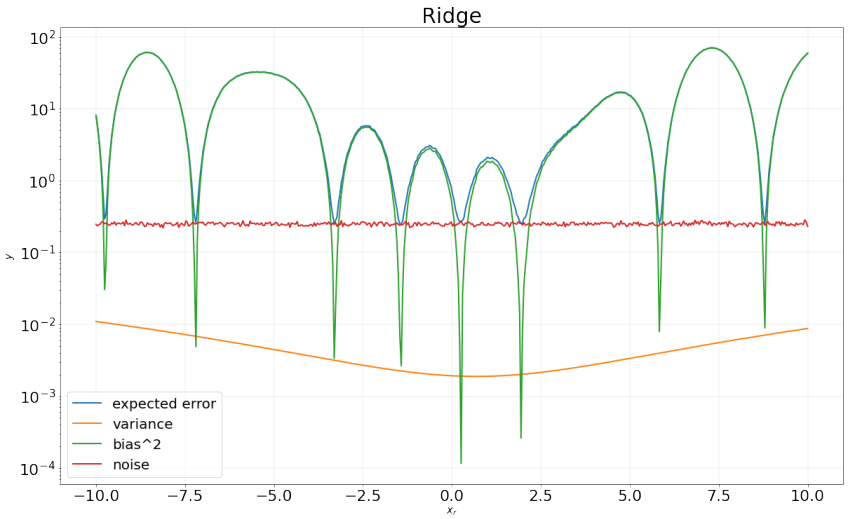
(i) For each x_i, compute its noise, its bias^2 and variance and store them respectively in 3 vectors N, B and V

(ii) We get that the mean of the noise, the bias^2 and the variance are respectively equal to the mean of N, of B and of V.

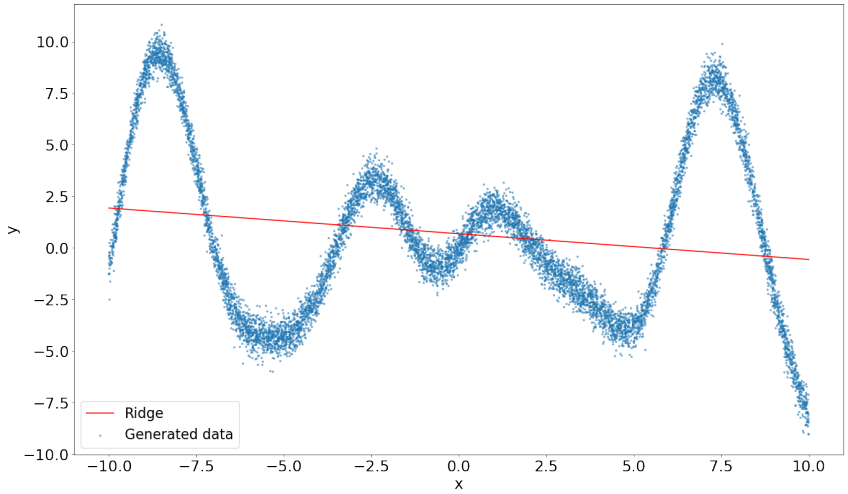
(c) The protocol will not be that viable because we rely mainly on the fact that we can generate as much y_{x0} given x_0 and without this we would not be able to have a good approximation of $E_{y|x0}[.]$. Therefore the *residual error and bias*^{2} could not be computed (if we follow the protocol). Only the *variance* could be measure but it if the data set is too small trained model will not be that relevant as they will underfit.

(d) Script for that part is *src/2d.ipynb* and the protocol is implemented as a Class on the file *src/python/protocol.py*

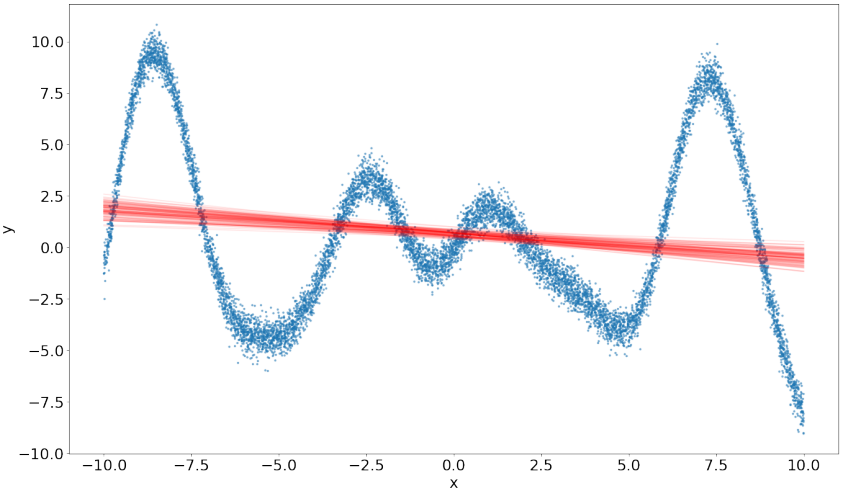
- **Linear - Ridge**



(a) Estimation of the noise, the squared bias, the variance and the expected error as a function of x using ridge regression, $\alpha = 1$



(b) Example of a trained ridge regression on 10k samples, with α=1



(c) Example of 100 trained ridge regression on 1000 different samples each, with α=1

Concerning ridge as expected results are not good, indeed it just draw a line while data generated are not linear at all

- ***bias*^{2}**: In order to explain what happened lets use an example, **6b** show a ridge regression trained on 10k samples, we can really see a correlation between the bounce and drop of the $bias^2$ and the curve generated by the data generator. Indeed when the curve move away from the straight line, its create a bounce of the $bias^2$, and when it get close to the line the $bias^2$ reduce really fast. This show that using a linear model may not be the best solution.
- **noise (residual error)**: As the noise is completely independent from the input x it remains quite constant, with a certain variance.
- **variance**: For this using an example (**6c**) will also help. We can see the different ridge regression as a seesaw, for which a slight different slope will cause an higher difference at extremities than at the center. This is what happen here and why the variance is higher at extremities than at the center as shown in **6a**. Having different slope is due to the fact that they are fed with different learning samples, therefore results differs slightly. In a more general view, as each model are fed with a large sample size the result from one and another will be quite close and this is why the variance is low. In the opposite if we would fed them with small LS size the variance would be higher.
- **expected error**: it is quite close to the squared bias. Which is expected as a linear model tend to have an higher bias than non-linear ones, due to the fact that it makes the assumption of a linear dependence of x and y.

It is worth mentioning that having a low *variance* and *high bias* leads to **underfitting**, which is clearly the case here as the model is not enough *complex*.

- **Non-linear - KNN**

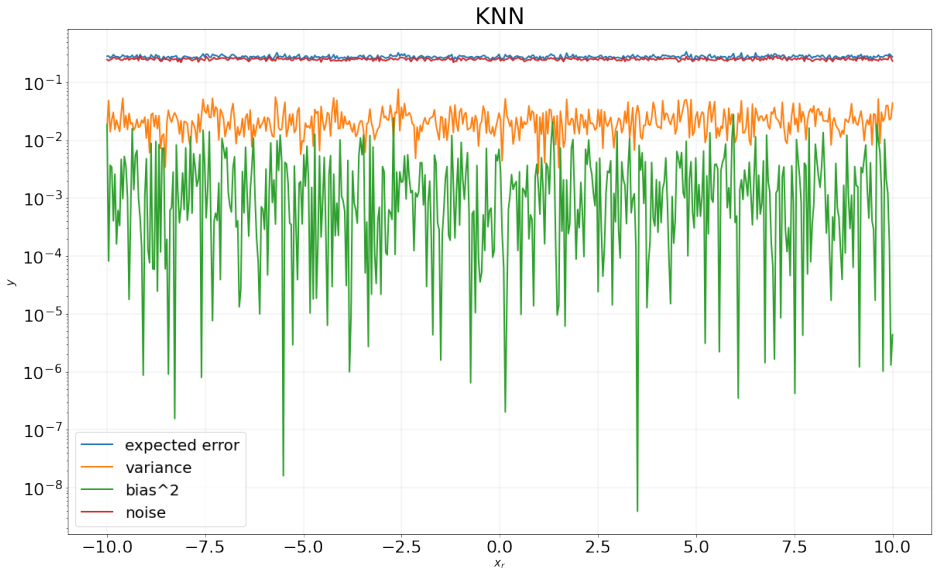


Figure 7. Estimation of the noise, the squared bias, the variance and the expected error as a function of x using KNN, $K = 10$

Using a non linear model give better results, indeed even if it has in average an higher **variance** than linear one the **expected error** is really close to the noise of the data (the minimal attaigable error). This is mainly due to the squared bias which is low (really low compare to the ridge one). However a low bias and high variance may lead to overfitting, but here the variance is still quite low therefor it seems to generalize quite well (see **0.2.e)**. We can conclude that **KNN** seems suited for this problem as it perform well when fed with a large samples size.

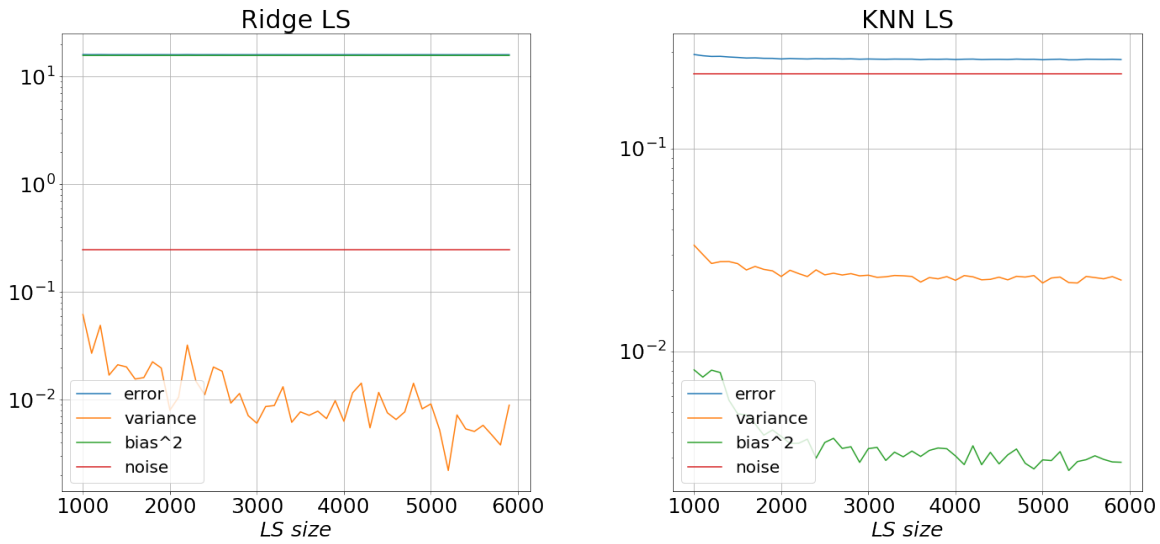
(e) (see *src/2e_*.ipynb for scripts*) The following measurements are made with the following default values if the graph is not a function of them:

(a) Number of LS: 10

(b) Sample size per LS: 1000

(c) Number of neighbors: 10

(d) Ridge coefficient α: 1.0



(a) $\alpha = 1.0$ (b) $k = 10$

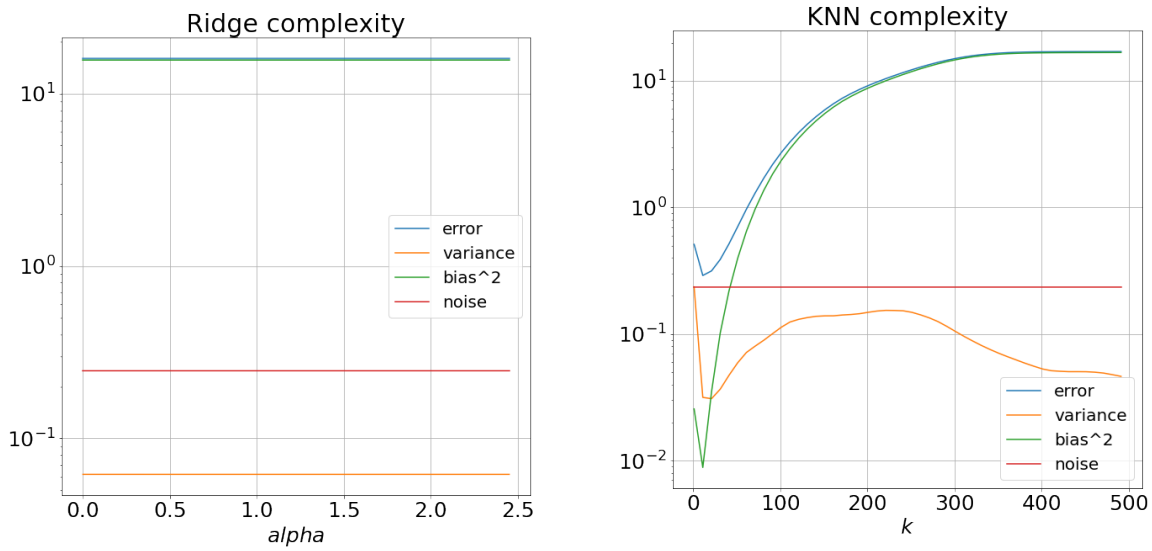
Figure 8. Error, variance, bias^2 and noise as a function of sample size per learning set

The noise remains constant and is independent of the learning set size. This is logical because it represents the noise in the data.

Regarding the squared bias, it is constant for the Ridge regression. This is due to the fact that our model is underfitting the regression problem: a model is underfitting when the sample size doesn't improve the regression. Therefore, KNN regression seems to be a good candidate in order to address our regression problem. The bias decreases very quickly as the sample size grows, we almost get a null bias. Intuitively, KNN Regressor is better when the LS size grows because the more samples we have for our model, the most probably we get samples that are closer to the point we wish to estimate.

Now for the variance, it is decreasing for both models. This is quite logic because for small datasets, the variability between two dataset is bigger and we will get that our models will make very differents predictions because they trained on not enough data. With big datasets, our models will likely train on the same data (because not too much variability between two datasets) and they will predict the same value (or at least very close) for a given input.

We can conclude that the expected error for the Ridge regression is independant of the learning set size while it will decrease for the KNN regression.



(a) Complexity decrease with k

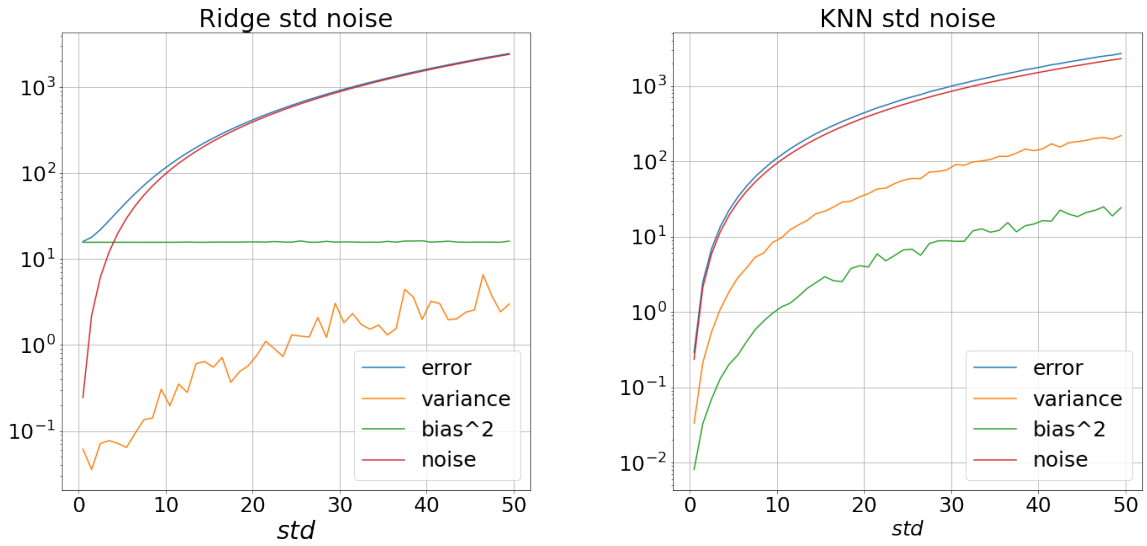
Figure 9. Error, variance, bias^2 and noise as a function of the model complexity

Again, the residual error is only dependent on the noise in the data and therefore it is independent from the complexity of the model (it is even independent from the model we use).

Regarding the squared bias, it remains constant for the Ridge regressor. Again, this proves that Ridge regression is not a good solution for our problem because it is not possible to tune its parameters in order to lower the bias. When it comes to KNN, we see that the bias increases with respect to the number of neighbors. Recall that each model is trained with a sample size of 1000 and in order to get a good prediction, we should consider points that are very close to the point we wish to estimate. Therefore, when the number of neighbors becomes too big, it takes into account too many points in the dataset that may be irrelevant for this prediction, but they do contribute to it.

The variance remains constant for the Ridge regressor, because again, this model doesn't improve in any case. KNN's variance decreases with respect to the number of neighbors. Assuming that 1000 samples is enough to describe the data, the averaged value when taking into account many neighbors is likely to be the same from one LS to another.

To conclude, KNN with a small number of neighbours (complex enough) is suitable



(a) $\alpha = 1.0$ (b) $k = 10$

Figure 10. Error, variance, bias^2 and noise as a function of the std of the noise

for our regression problem: The expected error is very close to 0.

We can see that both models behave the same way when we increase the standard deviation of the noise. Both models become very bad. Everything increases and the expected error is just very high.

For the noise, this is logic since by definition, it represents the noise in the data.

Regarding the bias, this is due to the fact that the model trains on very poor data and there is not much correlation anymore between the input x and the output y.

Now for the variability, this is also normal to see an increase of it between two learning sets because the noise is dominating over the correlation between the input and the output.

To conclude, both models are not suitable for a regression problem with very noisy data. In general, when there is too much noise in the data, we can not do much with it: there is no suitable regression method in this case.