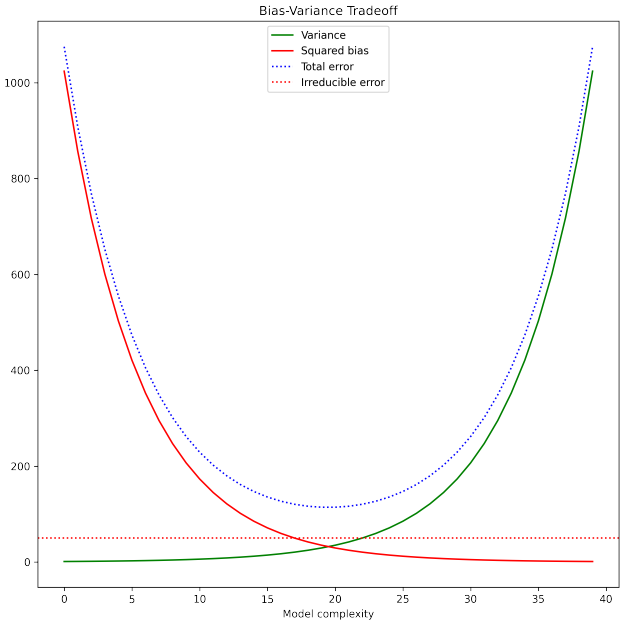
Machine Learning:
Assignment Sheet #8

Due on April 5, 2022 at 10:00

Group HB
Henri Sota, Enis Mustafaj

Problem 8.1

In this exercise, we repeat the general idea of the bias-variance trade-off for growing model complexity. To this end, you are asked to provide one plot that gives curves for the irreducible error, the bias, the variance, and the total (expected) generalization error for growing model complexity. *(You might have seen such a plot in the lecture material.)* In addition, comment in your own words the "behavior" of the four different curves. Again, in your own words, explain, in connection to these curves, the idea of the bias-variance trade-off.



In the plot above, we can see the relationship between (squared) bias, variance, irreducible error and total (expected) generalization error.

- The irreducible error is constant, independent of the model complexity, due to random noise. It is inherent to the collected data. The irreducible error can represent noise coming from collection of data, from the non-deterministic behavior of the phenomenon being modelled, and generally, any type of noise that cannot be easily defined.

- The bias error arises from erroneous assumptions in the learning algorithm of the model. It represents the difference between the average of the predicted value over different realizations of training data and the true value. A model with high bias oversimplifies the underlying model from which the data has been generated from and it misses to predict complex and relevant relations between the input and output. This is known as *underfitting*. The bias error curve typically has a high value for low model complexity and low value for high model complexity. This arises from the fact that with increased model complexity, the model is able to make more assumptions about the data and to capture the complex relations in the data.

- The variance error arises from sensitivity to small fluctuations in the training set. It represents the mean squared deviation of the predicted value from its expected value over different realizations of training data. A model with high variance fits to the training data quite well, but it suffers in other cases due to its inability to generalize beyond the training data. This is known as *overfitting*. The

variance curve typically has a low value for low model complexity and high value for high model complexity. This arises from the fact that with increased model complexity, the model starts to fit the training data too well with the aim to minimize the training error. As a result, the model will show different performance with similar datasets.

- The total (expected) generalization error is the sum of the irreducible error, the bias error and the variance error. Therefore its behavior depends on the behavior of the other curves.

The bias-variance trade-off is based on the choice of increasing or decreasing the model complexity to lower bias (increase variance) or lower variance (increase bias) with the aim to reduce the total (expected) generalization error. The model should accurately capture the underlying model in its training data while generalizing well to data that it has not seen before. It is typically impossible to do both simultaneously. Models with high bias produce simpler models that may underfit the data. On the other side, models with high-variance may overfit the data and perform poorly in different scenarios. Therefore, the trade-off lies in adjusting the model used with the aim to reduce both errors theoretically.

Problem 8.2

Recall from the lecture that the general bias-variance decomposition is given by

$$EGE(f, \mathbf{x}_0) = \sigma_\varepsilon^2 + [\mathrm{E}_T[f_T(\mathbf{x}_0)] - f_{exact}(\mathbf{x}_0)]^2 + \mathrm{E}_T\left[(f_T(\mathbf{x}_0) - \mathrm{E}_T[f_T(\mathbf{x}_0)])^2\right]$$

While in the lecture notes, we discuss this decomposition more concretely for kNN regression, we do not discuss it for linear regression by least squares. Since deriving the resulting decomposition for linear regression by least squares is a bit involved, we will not do this here. Instead, we want to compute the bias and variance term for concretely given data and the linear model trained via the least squares estimator.
To achieve this, we first assume to have $f_{exact}(x) = x^2$. Moreover, we consider the four samples $\mathcal{T}_1 = \{x_i^{(1)}, y_i^{(1)}\}_{i=1}^3$, $\mathcal{T}_2 = \{x_i^{(2)}, y_i^{(2)}\}_{i=1}^3$, $\mathcal{T}_3 = \{x_i^{(3)}, y_i^{(3)}\}_{i=1}^3$, $\mathcal{T}_4 = \{x_i^{(4)}, y_i^{(4)}\}_{i=1}^3$ from T with $N = 3$ training samples. These are given as follows

t	i	$x_i^{(t)}$	$\varepsilon_i^{(t)}$	$y_i^{(t)} = f_{exact}(x_i^{(t)}) + \varepsilon_i^{(t)}$
1	1	-1.0	-0.3	
1	2	-1.5	0.2	
1	3	0.5	0.1	
2	1	-0.5	0.0	
2	2	1.5	-0.2	
2	3	-1.5	0.3	
3	1	1.5	-0.1	
3	2	2.0	0.0	
3	3	-0.5	0.1	
4	1	1.0	-0.2	
4	2	-2.0	0.1	
4	3	1.5	0.3	

Finally we choose $x_0 = 0$.

a) Compute the output samples of the given training sets, i.e. fill in the remaining cells in the above table.

After the calculations, the output samples will be:

t	i	$x_i^{(t)}$	$\varepsilon_i^{(t)}$	$y_i^{(t)} = f_{exact}(x_i^{(t)}) + \varepsilon_i^{(t)}$
1	1	-1.0	-0.3	0.7
1	2	-1.5	0.2	2.45
1	3	0.5	0.1	0.35
2	1	-0.5	0.0	0.25
2	2	1.5	-0.2	2.05
2	3	-1.5	0.3	2.55
3	1	1.5	-0.1	2.15
3	2	2.0	0.0	4
3	3	-0.5	0.1	0.35
4	1	1.0	-0.2	0.8
4	2	-2.0	0.1	4.1
4	3	1.5	0.3	2.55

b) Now that you have all four training sets at hand, compute an estimator for the bias term. To this end, you first build for all four different training sets the linear model using linear regression by least squares, hence you obtain models $f_{\mathcal{T}_1}, f_{\mathcal{T}_2}, f_{\mathcal{T}_3}, f_{\mathcal{T}_4}$. You then need to estimate the expectation, recalling that for a random variable Z an estimator for its mean is given by

$$\mathrm{E}(Z) \approx \bar{Z} = \frac{1}{M} \sum_{i=1}^{M} z_i$$

where the $z_1, \ldots, z_M$ are M samples drawn from that random variable. *Hint: Use a computer to find the necessary partial results.*

The final training sets are:

$$\mathcal{T}_1 = \{(-1, 0.7), (-1.5, 2.45), (0.5, 0.35)\}$$

$$\mathcal{T}_2 = \{(-0.5, 0.25), (1.5, 2.05), (-1.5, 2.55)\}$$

$$\mathcal{T}_3 = \{(1.5, 2.15), (2, 4), (-0.5, 0.35)\}$$

$$\mathcal{T}_4 = \{(1, 0.8), (-2, 4.1), (1.5, 2.55)\}$$

By using the least squares formula:

$$\beta = (\mathcal{X}^T \cdot \mathcal{X})^{-1} \cdot \mathcal{X} \cdot y$$

we calculate the coefficients vector of the model for each training set:

- $\mathcal{T}_1$:

$$\beta = \left(\begin{pmatrix} 1 & 1 & 1 \\ -1 & -1.5 & 0.5 \end{pmatrix} \cdot \begin{pmatrix} 1 & -1 \\ 1 & -1.5 \\ 1 & 0.5 \end{pmatrix} \right)^{-1} \cdot \begin{pmatrix} 1 & 1 & 1 \\ -1 & -1.5 & 0.5 \end{pmatrix} \cdot \begin{pmatrix} 0.7 \\ 2.45 \\ 0.35 \end{pmatrix}$$

$$\beta = \begin{pmatrix} 0.59230769 \\ -0.86153846 \end{pmatrix}$$

  The predictor for $\mathcal{T}_1$ is $f_{\mathcal{T}_1}(x) = 0.59230769 - 0.86153846 \cdot x$ and the predicted value at x_0 is $f_{\mathcal{T}_1}(x_0) = 0.5923076923076922$

- $\mathcal{T}_2$:

$$\beta = \left(\begin{pmatrix} 1 & 1 & 1 \\ -0.5 & 1.5 & -1.5 \end{pmatrix} \cdot \begin{pmatrix} 1 & -0.5 \\ 1 & 1.5 \\ 1 & -1.5 \end{pmatrix} \right)^{-1} \cdot \begin{pmatrix} 1 & 1 & 1 \\ -0.5 & 1.5 & -1.5 \end{pmatrix} \cdot \begin{pmatrix} 0.25 \\ 2.05 \\ 2.55 \end{pmatrix}$$

$$\beta = \begin{pmatrix} 1.61428571 \\ -0.01428571 \end{pmatrix}$$

  The predictor for $\mathcal{T}_2$ is $f_{\mathcal{T}_2}(x) = 1.61428571 - 0.01428571 \cdot x$ and the predicted value at x_0 is $f_{\mathcal{T}_2}(x_0) = 1.6142857142857139$

- $\mathcal{T}_3$:

$$\beta = \left(\begin{pmatrix} 1 & 1 & 1 \\ 1.5 & 2 & -0.5 \end{pmatrix} \cdot \begin{pmatrix} 1 & 1.5 \\ 1 & 2 \\ 1 & -0.5 \end{pmatrix} \right)^{-1} \cdot \begin{pmatrix} 1 & 1 & 1 \\ 1.5 & 2 & -0.5 \end{pmatrix} \cdot \begin{pmatrix} 2.15 \\ 4 \\ 0.35 \end{pmatrix}$$

$$\beta = \begin{pmatrix} 0.86666667 \\ 1.3 \end{pmatrix}$$

  The predictor for $\mathcal{T}_3$ is $f_{\mathcal{T}_3}(x) = 0.86666667 + 1.3 \cdot x$ and the predicted value at x_0 is $f_{\mathcal{T}_3}(x_0) = 0.8666666666666669$

- $\mathcal{T}_4$:

$$\beta = \left(\begin{pmatrix} 1 & 1 & 1 \\ 1 & -2 & 1.5 \end{pmatrix} \cdot \begin{pmatrix} 1 & 1 \\ 1 & -2 \\ 1 & 1.5 \end{pmatrix} \right)^{-1} \cdot \begin{pmatrix} 1 & 1 & 1 \\ 1 & -2 & 1.5 \end{pmatrix} \cdot \begin{pmatrix} 0.8 \\ 4.1 \\ 2.55 \end{pmatrix}$$

$$\beta = \begin{pmatrix} 2.59534884 \\ -0.67209302 \end{pmatrix}$$

The predictor for $\mathcal{T}_4$ is $f_{\mathcal{T}_4}(x) = 2.59534884 - 0.67209302 \cdot x$ and the predicted value at x_0 is $f_{\mathcal{T}_4}(x_0) = 2.595348837209302$

By using the formula above, we calculate the expected value of $f_T(0)$:

$$E_T[f_T(0)] \approx \frac{1}{4} \cdot \sum_{i=1}^{4} f_{\mathcal{T}_i}(0) = 1.4171522276173438$$

Now we can calculate the bias:

$$bias = \mathrm{E}_T[f_T(\mathbf{x}_0)] - f_{exact}(\mathbf{x}_0) = 1.4171522276173438$$

c) Finally estimate the variance term. Here, it is useful to further observe that the variance term is indeed the variance of $f_T(\mathbf{x}_0)$ with respect to the random variable T, hence

$$\mathrm{E}_T\left[(f_T(\mathbf{x}_0) - \mathrm{E}_T[f_T(\mathbf{x}_0)])^2\right] = \mathrm{Var}_T(f_T(\mathbf{x}_0))$$

Then use that an unbiased estimator for the variance of a random variable Z is given by

$$\mathrm{Var}(Z) \approx \frac{1}{M-1} \sum_{i=1}^{M} (z_i - \bar{Z})^2$$

Hint: Use a computer to find the necessary partial results.

By using the formula above, we calculate the variance:

$$\mathrm{Var}_T(f_T(\mathbf{x}_0)) \approx \frac{1}{3} \cdot \sum_{i=1}^{4} (f_{\mathcal{T}_i}(0) - E_T[f_T(0)]) = 0.8034705742217966$$

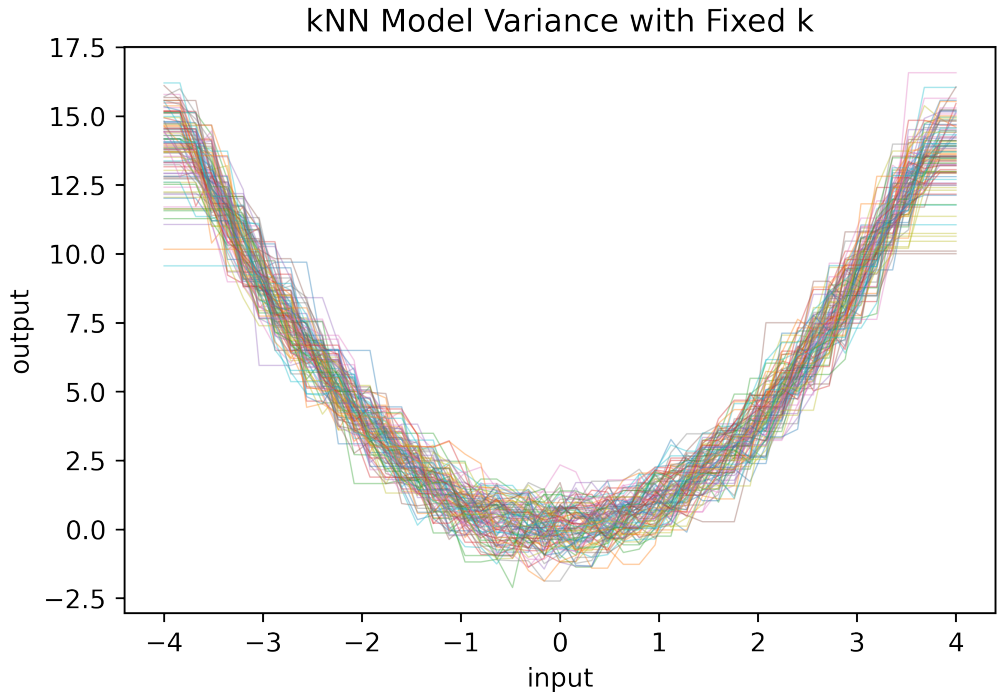
Programming Problem 8.1

In this programming exercise, we further study Example 7.2 from the lecture notes with its associated implementation. If we more carefully read the example, we observe that in the process of the evaluation of the different error contributions, we build $N_T = 100$ different kNN regression predictors for $k = 1, \ldots, 20$. Note however, that these predictors are only built from the training data $\{(x_i, f_{exact}(x_i)\}_{i=1}^{N}$ and not from $\{(x_i, f_{exact}(x_i) + \varepsilon_i)\}_{i=1}^{N}$.

a) Start by extending the existing implementation such that you also build all these predictors for the training data with the additional noise term.

The implementation has been extended and can be found in the file `programming_exercises.ipynb`.

b) Now, fix $k = 3$ and add a plot that contains the evaluation of all these predictors (with included noise term). Thus you obtain a total of $N_T = 100$ different curves. Try to use some appropriate coloring and transparency to make them properly visible.
The plot that you will then have created is a visualization of the variance of the predictor.



The code for generating this plot can be found in the file `programming_exercises.ipynb`.