

Introduction to Machine Learning

Project 2 - Bias and variance analysis

DOMINIKA PIECHOTA and PAWEŁ DOROSZ

1 Analytical derivations

Let us consider a regression problem, where each example $(x, y) \in \mathbb{R}^p \times \mathbb{R}$ is drawn i.i.d. from a distribution characterized as follows:

- The input x is drawn from a density $p(x)$;
- $y = h(x) + \varepsilon$, with h a function from \mathbb{R}^p to \mathbb{R} ;
- ε_i is drawn from a normal distribution $\mathcal{N}(0, \sigma^2)$.

Given a learning sample $LS = \{(x_1, y_1), \dots, (x_N, y_N)\}$ of N pairs, let us denote by $LS_x = \{x_1, \dots, x_N\}$ the inputs of the learning sample examples and by $LS_y = \{y_1, \dots, y_N\}$ their outputs. We consider in this question estimators \hat{f}_{LS} that are linear in the y_i , i.e. such that:

$$\hat{f}_{LS}(x) = \sum_{i=1}^N w_i(x; LS_x) y_i, \quad (1)$$

where the weights $w_i(x; LS_x)$ only depend on x and the inputs in LS_x .

1.1

Show that both linear regression and the k-nearest-neighbors (kNN) method fit into this class of estimators.

For the linear regression:

$$\hat{f}(x) = x^T \hat{\beta}$$

where (X is matrix $N \times p$):

$$\begin{aligned} X \hat{\beta} &= Y \\ X^T X \hat{\beta} &= X^T Y \\ \hat{\beta} &= (X^T X)^{-1} X^T Y \end{aligned}$$

Substituting:

$$\hat{f}(x) = x^T (X^T X)^{-1} X^T Y$$

This is multiplying: $x^T (X^T X)^{-1} X^T \times Y$ (vector $1 \times N$ and vector $N \times 1$).

Then (scalar product):

$$\hat{f}(x) = \sum_{i=1}^N [x^T (X^T X)^{-1} X^T]_i y_i$$

Weights $w_i(x; LS_x) = [x^T (X^T X)^{-1} X^T]_i$ depend only on x i LS_x . ■

For k-nearest-neighbor method:

$$\hat{f}(x) = \frac{1}{k} \sum_{i \in N_k(x)} y_i \quad (N_k(x) - \text{set of indices of the } k \text{ nearest neighbors of } x \text{ in } LS_x)$$

This can be written as:

$$\hat{f}(x) = \sum_{i=1}^N w_i(x; LS_x) y_i$$

where

$$w_i(x; LS_x) = \begin{cases} \frac{1}{k} & \text{if } x_i \text{ is among the k-NN of } x \\ 0 & \text{otherwise} \end{cases}$$

The weights depend only on x and LS_x (distances between points), not on y_i . ■

1.2

Explain why regression trees do not fit into this class, despite the fact that all their predictions are averages of outputs y_i of learning sample examples.

Despite the fact that regression tree predictions are averages of outputs y_i from the learning sample, they do **not** belong to the class of estimators linear in y_i because:

Even if the final prediction is $\hat{f}(x) = \sum_{i=1}^N w_i(x; LS_x) y_i$, which points belong to the leaf depend on LS_y , so the weights are not independent of y_i , because the structure of the tree depends on the outputs.

The leaf that contains point x is determined by the sequence of splits in the tree, and these splits are chosen based on minimizing variance y_i in the nodes and minimizing impurity.

Example: Minimizing the Sum of Squared Errors (SSE) in the child nodes:

$$SSE = \sum_{i \in R_1} (y_i - \bar{y}_{R_1})^2 + \sum_{i \in R_2} (y_i - \bar{y}_{R_2})^2$$

where:

- R_1, R_2 - regions (child nodes) after the split
- $\bar{y}_{R_1}, \bar{y}_{R_2}$ - mean values of y in the regions

$$\bar{y}_{R_1} = \frac{1}{|R_1|} \sum_{i \in R_1} y_i, \quad \bar{y}_{R_2} = \frac{1}{|R_2|} \sum_{i \in R_2} y_i$$

This depends on the specific values of y_i in the data.

Conclusion: Given the same LS_x but different LS_y , the learned tree structure (and thus the weights $w_i(x; LS_x)$) would be different, demonstrating the dependence on y_i values. ■

1.3

Decompose the following expected conditional mean square error into a residual error, a bias and a variance term at a fixed point x_0 :

$$\mathbb{E}_{LS_y|LS_x} \left\{ \mathbb{E}_{y|x_0} \left[(y - \hat{f}_{LS}(x_0))^2 \right] \right\}. \quad (2)$$

This decomposition takes thus into account only the variability coming from the outputs of the learning sample examples.

Hint: adapt the decomposition from the course to the fact that expectations over \mathbb{E}_{LS} are replaced by expectations over $\mathbb{E}_{LS_y|LS_x}$, and then plug (1) into the resulting terms. In the end, the different terms should be expressed using the problem and method parameters only and not contain any expectation.

Decomposition of the expected conditional mean square error. We want to decompose the following expected conditional mean square error at a fixed point x_0 :

$$E = \mathbb{E}_{LS_y|LS_x} \left\{ \mathbb{E}_{y|x_0} \left[(y - \hat{f}_{LS}(x_0))^2 \right] \right\}$$

(1) **Introduce the Bayes model**

$$h_B(x_0) = \mathbb{E}_{y|x_0} \{y\}$$

(2) **Add and subtract $h_B(x_0)$**

$$\begin{aligned} (y - \hat{f}_{LS}(x_0))^2 &= (y - h_B(x_0) + h_B(x_0) - \hat{f}_{LS}(x_0))^2 \\ &= (y - h_B(x_0))^2 + (h_B(x_0) - \hat{f}_{LS}(x_0))^2 + 2(y - h_B(x_0))(h_B(x_0) - \hat{f}_{LS}(x_0)) \end{aligned}$$

(3) **Compute $\mathbb{E}_{y|x_0}$**

$$\mathbb{E}_{y|x_0} \{(y - h_B(x_0))^2\} = \sigma^2 \quad (\text{noise})$$

$$\mathbb{E}_{y|x_0} \{(h_B(x_0) - \hat{f}_{LS}(x_0))^2\} = (h_B(x_0) - \hat{f}_{LS}(x_0))^2 \quad (\text{due to independence from } y)$$

$$\mathbb{E}_{y|x_0} \{2(y - h_B(x_0))(h_B(x_0) - \hat{f}_{LS}(x_0))\} = 0 \quad (\text{due to the definition of } h_B(x_0) \text{ and})$$

the fact that the components of the product are independent of each other)

Thus:

$$\mathbb{E}_{y|x_0} \left\{ (y - \hat{f}_{LS}(x_0))^2 \right\} = \sigma^2 + (h_B(x_0) - \hat{f}_{LS}(x_0))^2$$

(4) **Compute $\mathbb{E}_{LS_y|LS_x}$**

$$E = \mathbb{E}_{LS_y|LS_x} \left\{ \sigma^2 + (h_B(x_0) - \hat{f}_{LS}(x_0))^2 \right\} = \sigma^2 + \mathbb{E}_{LS_y|LS_x} \left\{ (h_B(x_0) - \hat{f}_{LS}(x_0))^2 \right\}$$

(5) Bias-variance decomposition

The average model:

$$\bar{f}(x_0) = \mathbb{E}_{LS_y|LS_x} \{\hat{f}_{LS}(x_0)\}$$

Then:

$$\begin{aligned} & \mathbb{E}_{LS_y|LS_x} \{(h_B(x_0) - \hat{f}_{LS}(x_0))^2\} \\ &= \mathbb{E}_{LS_y|LS_x} \{(h_B(x_0) - \bar{f}(x_0) + \bar{f}(x_0) - \hat{f}_{LS}(x_0))^2\} \\ &= (h_B(x_0) - \bar{f}(x_0))^2 + \mathbb{E}_{LS_y|LS_x} \{(\bar{f}(x_0) - \hat{f}_{LS}(x_0))^2\} + 2(h_B(x_0) - \bar{f}(x_0)) \cdot \underbrace{\mathbb{E}_{LS_y|LS_x} \{\bar{f}(x_0) - \hat{f}_{LS}(x_0)\}}_{=0} \\ & (E_{LS_y|LS_x} \{(h_B(x_0) - \hat{f}_{LS}(x_0))^2\} = (h_B(x_0) - \hat{f}_{LS}(x_0))^2 \text{ because does not depend on LS}) \end{aligned}$$

Final decomposition:

$$E = \underbrace{\sigma^2}_{\text{noise}(x_0)} + \underbrace{(h_B(x_0) - \bar{f}(x_0))^2}_{\text{bias}^2(x_0)} + \underbrace{\mathbb{E}_{LS_y|LS_x} \{(\bar{f}(x_0) - \hat{f}_{LS}(x_0))^2\}}_{\text{variance}(x_0)}$$

Same expression but using method parameters:

Since $\hat{f}_{LS}(x_0) = \sum_{i=1}^N w_i(x_0; LS_x) y_i$ and $y_i = h(x_i) + \epsilon_i$ with $\epsilon_i \sim N(0, \sigma^2)$:

$$\bar{f}(x_0) = \sum_{i=1}^N w_i(x_0; LS_x) h(x_i) \quad (1)$$

$$\text{variance}(x_0) = \sigma^2 \sum_{i=1}^N w_i^2(x_0; LS_x) \quad (2)$$

Thus the final expression is:

$$E = \sigma^2 + \left(h_B(x_0) - \sum_{i=1}^N w_i(x_0; LS_x) h(x_i) \right)^2 + \sigma^2 \sum_{i=1}^N w_i^2(x_0; LS_x)$$

(1) Why the average model has this form?

We have: $\hat{f}_{LS}(x_0) = \sum_{i=1}^N w_i(x_0; LS_x) y_i$

The average model is:

$$\bar{f}(x_0) = E_{LS_y|LS_x} \{\hat{f}_{LS}(x_0)\} = E_{LS_y|LS_x} \left\{ \sum_{i=1}^N w_i(x_0; LS_x) y_i \right\} \stackrel{*}{=} \sum_{i=1}^N w_i(x_0; LS_x) E_{LS_y|LS_x} \{y_i\}$$

(*) We can take weights w_i out of the expectation, because they depend only on LS_x (which is fixed).

But $E_{LS_y|LS_x} \{y_i\} = E_{y|x_i} \{y\} = h(x_i)$ (because $y_i = h(x_i) + \epsilon_i$ and $E\{\epsilon_i\} = 0$)

Therefore:

$$\bar{f}(x_0) = \sum_{i=1}^N w_i(x_0; LS_x) h(x_i)$$

(2) Why the variance has this form?

$$\text{variance}(x_0) = E_{LS_y|LS_x} \left\{ (\hat{f}_{LS}(x_0) - \bar{f}(x_0))^2 \right\}$$

(1) Write the difference

$$\hat{f}_{LS}(x_0) - \bar{f}(x_0) = \sum_{i=1}^N w_i(x_0; LS_x) y_i - \sum_{i=1}^N w_i(x_0; LS_x) h(x_i) = \sum_{i=1}^N w_i(x_0; LS_x) (y_i - h(x_i))$$

But $y_i - h(x_i) = \epsilon_i$ (noise), so:

$$\hat{f}_{LS}(x_0) - \bar{f}(x_0) = \sum_{i=1}^N w_i(x_0; LS_x) \epsilon_i$$

(2) Square it

$$(\hat{f}_{LS}(x_0) - \bar{f}(x_0))^2 = \left(\sum_{i=1}^N w_i(x_0; LS_x) \epsilon_i \right)^2 = \sum_{i=1}^N \sum_{j=1}^N w_i(x_0; LS_x) w_j(x_0; LS_x) \epsilon_i \epsilon_j$$

(3) Compute the expectation

$$\text{variance}(x_0) = E_{LS_y|LS_x} \left\{ \sum_{i=1}^N \sum_{j=1}^N w_i w_j \epsilon_i \epsilon_j \right\} = \sum_{i=1}^N \sum_{j=1}^N w_i w_j E\{\epsilon_i \epsilon_j\}$$

(4) Consider noise properties

- $\epsilon_i \sim N(0, \sigma^2)$
- ϵ_i are independent, so $E\{\epsilon_i \epsilon_j\} = E\{\epsilon_i\} \cdot E\{\epsilon_j\} = 0 \cdot 0$ for $i \neq j$
- $E\{(\epsilon_i)^2\} = E\{(\epsilon_i - 0)^2\} = \sigma^2$

Therefore:

$$E\{\epsilon_i \epsilon_j\} = \begin{cases} \sigma^2 & \text{if } i = j \\ 0 & \text{if } i \neq j \end{cases}$$

(5) Simplify the sum

$$\text{variance}(x_0) = \sum_{i=1}^N w_i^2(x_0; LS_x) \sigma^2 = \sigma^2 \sum_{i=1}^N w_i^2(x_0; LS_x)$$

1.4

Simplify further this decomposition in the case of the kNN model and use this result to discuss the effect of k on each term of the bias-variance decomposition.

k-NN Bias-Variance Decomposition

For k-NN, the weights are:

$$w_i(x_0; LS_x) = \begin{cases} \frac{1}{k} & \text{if } x^i \text{ is among the k-NN of } x_0 \\ 0 & \text{otherwise} \end{cases}$$

Simplified decomposition for k-NN:

$$\begin{aligned} \text{noise}(x_0) &= \sigma^2 \\ \text{bias}^2(x_0) &= \left(h_B(x_0) - \frac{1}{k} \sum_{i \in N_k(x_0)} h(x_i) \right)^2 \\ \text{variance}(x_0) &= +\sigma^2 \sum_{i=1}^N w_i^2(x_0; LS_x) = \sigma^2 \sum_{i=1}^N \left(\frac{1}{k}\right)^2 = \sigma^2 \sum_{i=1}^k \left(\frac{1}{k}\right)^2 = \frac{\sigma^2}{k} \end{aligned}$$

Thus the complete decomposition is:

$$E = \sigma^2 + \left(h_B(x_0) - \frac{1}{k} \sum_{i \in N_k(x_0)} h(x_i) \right)^2 + \frac{\sigma^2}{k}$$

Effect of k on each term:

- **Residual error (σ^2):** Independent of k - inherent data noise
- **Bias:** Generally increases with k
 - Small k: Low bias (adapts to local structure)
 - Large k: High bias (oversmoothing)
- **Variance:** Decreases with k as $\frac{1}{k}$
 - Small k: High variance (sensitive to data)
 - Large k: Low variance (stable predictions)

Bias-Variance Trade-off:

- **Small k:** Low bias, high variance → overfitting
- **Large k:** High bias, low variance → underfitting
- **Optimal k:** Balances bias and variance (found by cross-validation)

1.5

Let us now consider the regular bias-variance decomposition of the following expected error:

$$\mathbb{E}_{LS_x, LS_y} \left\{ \mathbb{E}_{y|x_0} \left[(y - \hat{f}_{LS}(x_0))^2 \right] \right\}. \quad (3)$$

Briefly discuss intuitively how the terms of the decomposition of this error relate to the corresponding terms of the decomposition of (2) averaged over LS_x (are they equal/smaller/greater relative to each other and why?).

Comparison of Error Decompositions

Conditional decomposition (2):

$$E^{(2)} = E_{LS_y|LS_x} \left\{ E_{y|x_0} \left[(y - \hat{f}_{LS}(x_0))^2 \right] \right\} = \sigma^2 + \text{bias}_{(2)}^2(x_0) + \text{var}_{(2)}(x_0)$$

Full decomposition (3):

$$E^{(3)} = E_{LS_x, LS_y} \left\{ E_{y|x_0} \left[(y - \hat{f}_{LS}(x_0))^2 \right] \right\} = \sigma^2 + \text{bias}_{(3)}^2(x_0) + \text{var}_{(3)}(x_0)$$

Detailed comparison of terms:

- **Residual error (σ^2):**
 - Equal in both decompositions: $\sigma_{(3)}^2 = \sigma_{(2)}^2$
 - Depends only on inherent data noise, independent of the learning sample
 - Represents irreducible error due to the randomness in y for fixed x_0
- **Bias squared:**
 - $\text{bias}_{(3)}^2(x_0) = E_{LS_x} \left\{ \text{bias}_{(2)}^2(x_0) \right\}$
 - Relationship: $\text{bias}_{(3)}^2(x_0)$ can be smaller or larger than $\text{bias}_{(2)}^2(x_0)$ for a specific LS_x
 - For a *lucky* LS_x configuration: $\text{bias}_{(2)}^2(x_0) < \text{bias}_{(3)}^2(x_0)$
 - For an *unlucky* LS_x configuration: $\text{bias}_{(2)}^2(x_0) > \text{bias}_{(3)}^2(x_0)$
 - $\text{bias}_{(3)}^2$ represents the average performance over all possible input configurations
 - $\text{bias}_{(2)}^2$ shows performance for a specific input configuration
- **Variance:**
 - $\text{var}_{(3)}(x_0) = E_{LS_x} \left\{ \text{var}_{(2)}(x_0) \right\} + \text{var}_{LS_x}(\bar{f}(x_0))$
 - Relationship: $\text{var}_{(3)}(x_0) \geq \text{var}_{(2)}(x_0)$ (always)
 - The full variance consists of two components:
 - (1) $E_{LS_x} \left\{ \text{var}_{(2)}(x_0) \right\}$: average of conditional variances (output noise)
 - (2) $\text{var}_{LS_x}(\bar{f}(x_0))$: variance due to different input configurations
 - $\text{var}_{(2)}$ captures only variability from output noise for fixed LS_x
 - $\text{var}_{(3)}$ captures both output noise AND input sampling variability

Key insights:

- The conditional decomposition (2) shows performance for a **specific learning sample**
- The full decomposition (3) shows the **expected performance** of the learning algorithm
- In practice, we usually care about (3) as it represents real-world expected performance
- For any specific LS_x , $E^{(2)}$ can be smaller or larger than $E^{(3)}$ depending on how lucky we are with the input configuration
- However, when averaged over LS_x , we always have: $E^{(3)} \geq E_{\text{average}}^{(2)}$

Final summary:

$$\sigma_{(3)}^2 = \sigma_{(2)}^2$$

$$\text{bias}_{(3)}^2 = E_{LS_x} \{\text{bias}_{(2)}^2\} \quad (\text{can be smaller or larger for specific } LS_x)$$

$$\text{var}_{(3)} = E_{LS_x} \{\text{var}_{(2)}\} + \text{var}_{LS_x}(\bar{f}) \geq \text{var}_{(2)} \quad (\text{always greater or equal})$$

The full error $E^{(3)}$ is generally more representative of real-world performance as it accounts for all sources of variability in the learning process.

2 Empirical analysis

Let us consider the same regression setting as in the previous section with:

- $p(x)$ such that each feature value x_j ($j = 1, \dots, p$) is drawn uniformly from $[-10, 10]$,
- $h(x) = \sin(2x_1) + x_1 \cos(x_1 - 1)$,
- $\sigma = 1$.

The output thus only depends on the first feature x_1 , while the other $p - 1$ features are said to be irrelevant for predicting the output.

Unless we ask you to study the impact of these parameters, you can use the following default values for the learning sample size and the total number of features: $N = 80$ and $p = 5$. We do not specify how to set all other parameter values or ranges on purpose. It is your responsibility to choose them wisely in order to demonstrate the expected behaviours.

2.1

Using all information you have about the problem, describe a protocol to estimate the residual error, the squared bias, and the variance at a given point x , and for a given supervised learning algorithm. Explain also how to compute an estimate of the mean values of the same quantities (over $p(x)$).

Protocol for estimating squared bias, variance, and residual error at point x :

- (1) Generate M independent training sets (LS) with N samples each,
- (2) For each set, train a selected regression model,
- (3) Predict the value at point x for each previously trained model ($\bar{pred}(x)$),
- (4) The average of the predictions gives an estimate of the expected prediction ($\bar{\bar{pred}}(x)$),
- (5) Calculate statistics such as squared bias, variance, residual and expected error:
 - $bias^2(x) = (\bar{pred}(x) - h(x))^2$
 - $variance(x) = \frac{1}{M}(\bar{pred}(x) - \bar{\bar{pred}}(x))^2$
 - $residual(x) = \sigma^2$
 - $expected\ error(x) = bias^2(x) + variance(x) + residual(x)$

To compute estimates of the mean values of these quantities assign values from the range $[-10, 10]$ to feature x_1 in sequence and check the residual, squared bias and variance results for each point, then take their average.

2.2

Implement and use this protocol to estimate the residual error, the squared bias, the variance, and the expected error for ridge regression, kNN, and regression trees for a test set of random points drawn from $p(x)$. Plot the resulting quantities as a function of the first feature x_1 only. Generate also a plot of the Bayes model and the average model of all three learning algorithms for the same set of inputs as a function again of x_1 only. Comment these results.

Plots of results and comment (for default $N = 80, p = 5$)

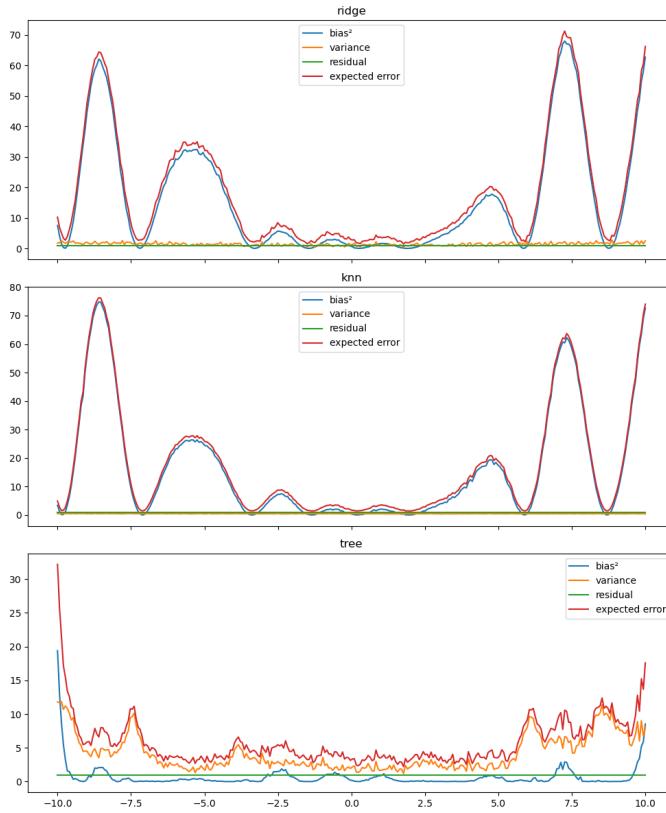


Fig. 1. Bias-variance decomposition

1. Ridge regression

As a linear method with regularization, the ridge model has limited flexibility and is unable to accurately map rapidly changing function shapes, what leads to:

- High squared bias - the model does not capture the complex structure of the function (particularly evident in ‘peaks’ and ‘troughs’)
- Very low variance - the model is stable and insensitive to data changes

The expected error is dominated by bias, as can be seen from the almost perfect coverage of the curves.

For such a non-linear function, ridge regression is too rigid and underfitted.

2. kNN method

The k-NN model is more flexible than ridge model but still smooths the data heavily when averaged over many samples or when k is relatively large.

- High squared bias - very similar to the ridge in many regions, meaning it cannot capture the complex structure of the function
- Low variance - surprisingly low variance, which suggests either a relatively large k or that the training noise is small.
- Expected error again dominated by bias rather than variance.

Conclusion: k-NN performs better than ridge in principle, but still fails to track rapid oscillations of the true function. It is flexible, but not flexible enough for this particular target.

3. Regression tree

Regression trees are highly flexible, capable of capturing local irregularities.

- Very low squared bias - regression trees follow the shape of the target function very well on each dataset.
- Very high variance - regression trees are extremely sensitive to small changes in training data, the variance curve dominates.
- Expected error is the smallest.

Conclusion: The regression tree has the smallest expected error so we consider it as the best model.

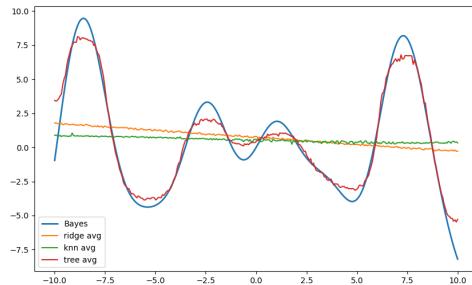


Fig. 2. Averaged models vs. Bayes

The Bayes curve represents the true underlying function, which is highly nonlinear.

- Ridge regression produces a nearly straight, weakly sloping line, showing clear underfitting – it cannot capture the nonlinear structure at all.
- k-NN also fails to follow the true function: its average prediction is over-smoothed and almost flat, indicating strong bias and insufficient flexibility for this problem.
- Regression tree is the closest to the Bayes curve. It follows the main peaks and valleys of the true function, although still with noticeable deviations. This reflects low bias, but some instability remains.

Overall, among the three models, only the regression tree captures the general nonlinear shape, while ridge and k-NN underfit significantly when averaged across datasets.

Plots of results and brief comment (for $N = 8000, p = 5$)

We chose these parameters to show that the data size $N = 80$ is definitely too small to train well performing models on given datasets. This applies especially to kNN, which gives reliable results with a significantly larger learning sample sizes.

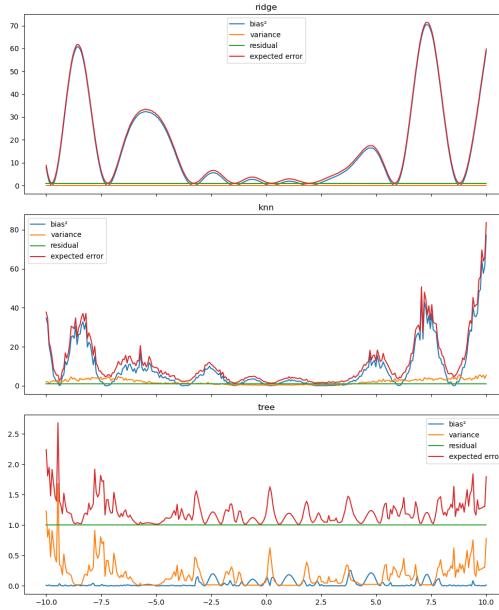


Fig. 3. Bias-variance decomposition $N=8000$

It can be seen that **ridge regression has not improved** significantly because it is still a linear model, so there is no chance of approximating a non-linear function well.

As for **kNN, there is a huge improvement** (the prediction ceases to be similar to a linear function and begins to be similar to a non-linear one), and the model fits much better.

Regression trees show a significant improvement, especially in terms of variance reduction.

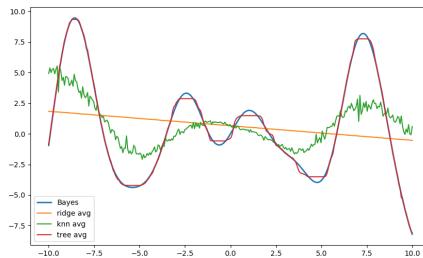


Fig. 4. Averaged models vs. Bayes $N=8000$

2.3

Use this protocol to estimate the mean values of the squared error, the residual error, the squared bias, and the variance for the same three regression methods as a function of:

- **the size of the learning set,**
- **the model complexity,**
- **the number of irrelevant variables.**

Comment your results and support your observations with the appropriate plots.

2.3.1. Function of size of the learning set.

- **Ridge Regression:** Bias remains consistently high, variance decreases slightly with larger N

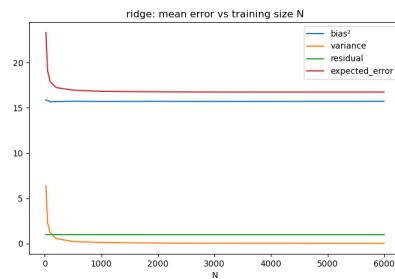


Fig. 5. N vs. mean error - Ridge regression

- **kNN:** Significant bias reduction with increasing N, variance shows modest improvement

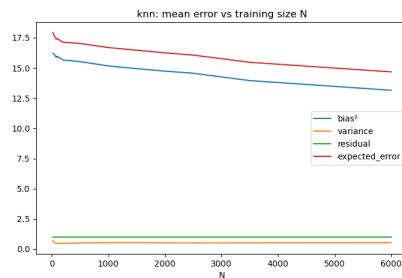


Fig. 6. N vs. mean error - kNN

- **Regression Trees:** Dramatic variance and bias reduction

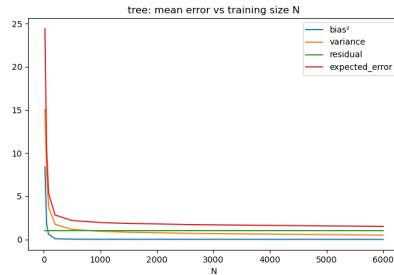


Fig. 7. N vs. mean error - Regression trees

Interpretation:

- **Ridge regression** is bias-dominated - more data doesn't help with model primitivity
- kNN benefits from density estimation - more neighbors available for better averaging and results
- **Regression trees** leverage additional data to build more stable splits, reducing overfitting

2.3.2.a Function of model complexity. (For default values N=80, p=5)

- **Ridge Regression (alpha):** On the whole interval of alpha we have high bias and low variance → underfitting pattern, due to the primitivity of the model

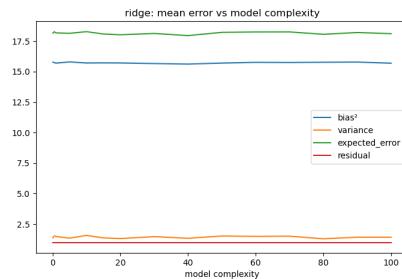


Fig. 8. Complexity vs. mean error - Ridge regression

- **kNN (k):** On the whole interval of k we have high bias due to small amount of points (N) in the learning sample. However, the variance decreases rapidly for initial k due to the reduction of overfitting through predicting by a larger number of neighbours.

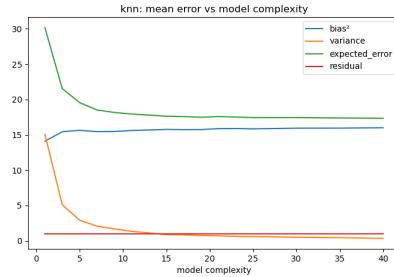


Fig. 9. Complexity vs. mean error - kNN

- **Regression Trees ($maxdepth$):**
 - Shallow trees: High bias and low variance → cannot capture complexity.
 - Deep trees: Low bias, higher variance → overfitting to training noise.
- This method performs best. Rapid decrease in bias and slight increase in variance with increasing tree depth.

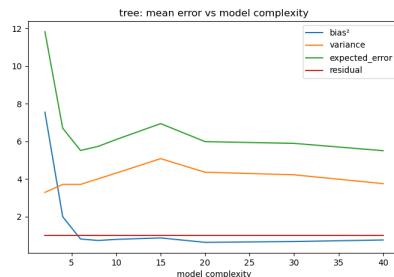


Fig. 10. Complexity vs. mean error - Regression trees

2.3.2.b Function of model complexity. (For values N=8000, p=5)

- **Ridge Regression (alpha):** On the whole interval of alpha we have high bias and low variance → underfitting pattern, due to the primitivity of the model

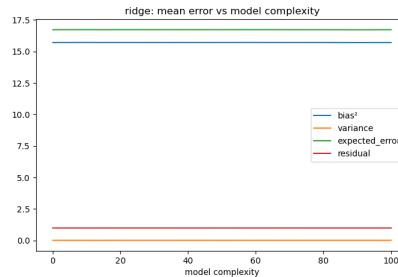


Fig. 11. Complexity vs. mean error - Ridge regression N=8000

- **kNN (k):** As k increases, we see an increase in bias caused by increasing underfitting. Classically, variance decreases, and we have an intersection point indicating the best trade-off between bias and variance.

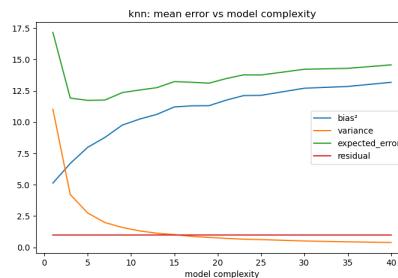


Fig. 12. Complexity vs. mean error - kNN N=8000

- **Regression Trees (*maxdepth*):** Regression trees improve their performance; we see decreasing bias and a decrease followed by an increase in variance as *maxdepth* increases, which allows us to conclude that this model is competent in this problem and to select the appropriate parameters. This method performs best.

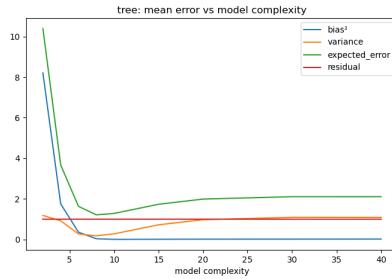


Fig. 13. Complexity vs. mean error - Regression trees N=8000

2.3.3.a Function of number of irrelevant variables. (For default values N=80, p=5)

- **Ridge Regression:** variance increases, gradual performance degradation

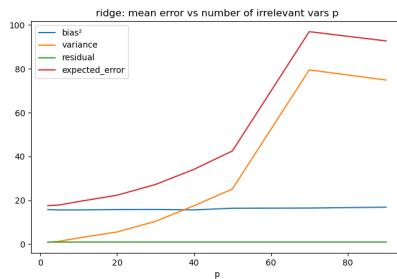


Fig. 14. p vs. mean error - Ridge regression

- **kNN:** almost no change in bias and variance

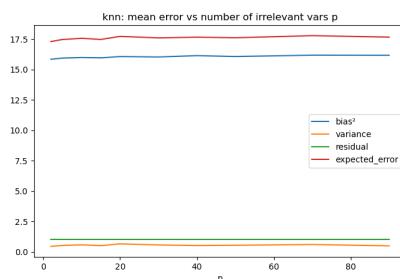


Fig. 15. p vs. mean error - kNN

- **Regression Trees:** bias and variance increase - curse of dimensionality

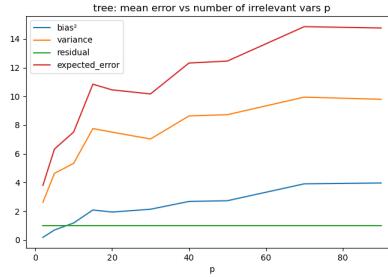


Fig. 16. p vs. mean error - Regression trees

Interpretation:

- **Ridge regression** spreads coefficients across all features, diluting the importance of x_1 , so the model is searching for patterns in other variables .
- **kNN** is trained for $N = 80 \rightarrow k$ is around 30 so it's no surprise that it has high bias and low variance, adding more dimensions does not worsen an already poor model.
For larger N , the model would likely worsen its performance as p increases, because the distances between points a and b would be measured equally in terms of all their coordinates. The weight of x_1 would decrease, which would cause the nearest neighbours to be selected almost entirely regardless of x_1 .
- **Regression trees** are also sensitive to the addition of dimensions and decide on worse splits, which are becoming less and less dependent on x_1 . Splits depend more and more on irrelevant variables so trees may look for patterns in them that do not quite exist, leading to high variance.

2.3.3.b Function of number of irrelevant variables. (For values N=8000, p=5)

- **Ridge Regression:** no change, ridge regression is still not sensitive with the nonlinear data.

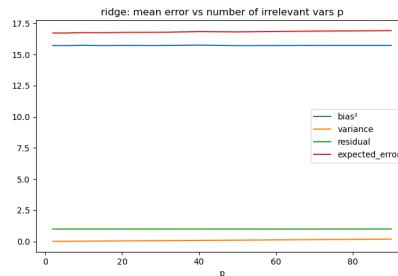


Fig. 17. p vs. mean error - Ridge regression N=8000

- **kNN:** it can be seen that for small p, the model achieves excellent results, which is due to the fact that the weight of the x_1 coordinate when measuring distance will be large.

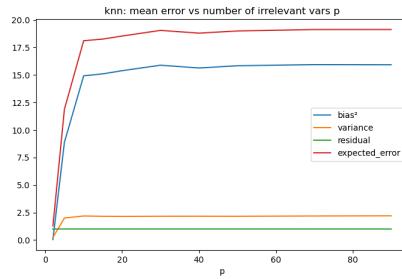


Fig. 18. p vs. mean error - kNN N = 8000

- **Regression Trees:** bias and variance remain very low because trees are very good at detecting non-linear trends when they are guaranteed a sufficiently large learning sample.

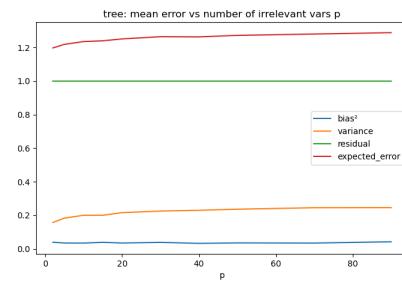


Fig. 19. p vs. mean error - Regression trees N = 8000

2.4

One generic method to reduce variance is bagging (for “bootstrap aggregating”), which consists in growing several models from bootstrap samples drawn from the original LS and then averaging their predictions. Apply this bagging idea on both linear regression, kNN, and regression trees, and evaluate its impact on bias and variance for increasing values of the number of models that are averaged. In particular, discuss the interest of applying bagging in combination with each method.

- **Ridge Regression with Bagging**

- Variance Reduction: Moderate improvement
- Bias Impact: No impact

Ridge already has low variance due to L2 regularization. Bagging provides additional stability through averaging.

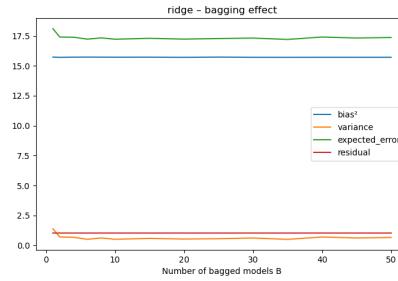


Fig. 20. Bagging - Ridge regression

- **kNN with Bagging:**

- Variance Reduction: Moderate improvement
- Bias Impact: No impact

Bagging effectively averages over different neighborhood selections. Bootstrap sampling creates diverse training sets for kNN. Particularly useful for kNN as it stabilizes the distance-based predictions

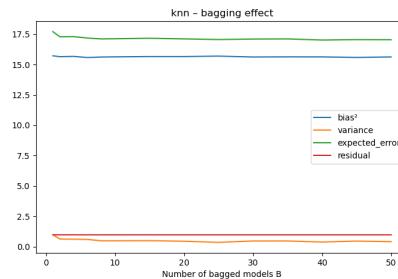


Fig. 21. Bagging - kNN

- **Regression Trees with Bagging:**

- Variance Reduction: Significant improvement
- Bias Impact: Negligible change

Trees are high-variance estimators due to their instability, so small changes in data can lead to completely different tree structures. Bagging leverages this instability to create diverse models. Each bootstrap sample produces trees that capture different aspects of the data and then the results are averaged.

The biggest improvement among all methods tested

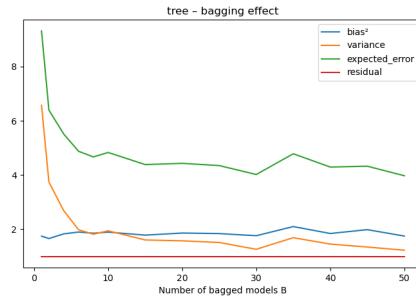


Fig. 22. Bagging - Regression trees

General Observation: Bagging consistently reduces variance across all methods, with the degree of improvement depending on the base algorithm's characteristics.

General Observation for N = 8000, p = 5: When N was increased to 8000, the results for ridge regression did not change. For kNN, a significant decrease in bias and a slight increase in variance can be observed compared to the results for N = 80. For regression trees, bias and variance reach almost zero, and the number of bagged models has negligible effect on the results.