ELEN0062 - Introduction to Machine Learning
Project 2 - Bias and variance analysis

October 23rd, 2024

The goal of this second assignment is to help you better understand the important notions of bias and variance. The first part is purely theoretical, while the second part requires to perform some experiments with Scikit-learn. You should hand in a *brief* report giving your developments, observations and conclusions along with the scripts you have implemented to answer the questions of the second part. The project must be carried out by groups of at most *three students* and submitted on Gradescope[1] before *November 22, 23:59 GMT+2*. There will be two projects to submit to: one for your python scripts and one for your report.

1 Analytical derivations

Let us consider a unidimensional regression problem $y = f(\mathbf{x}) + \epsilon$ where $\epsilon \sim \mathcal{N}(0, \sigma^2)$ and let $LS = \{(\mathbf{x}^1, y^1), \ldots, (\mathbf{x}^N, y^N)\}$ denote the learning sample (of fixed size N). To simplify the analysis, we further assume that the input values $\mathbf{x}^i$ of the N learning sample examples are fixed in advance, i.e., only their outputs y^i are random.

1. Show that the expected generalization error of the k-Nearest Neighbours algorithm at some point $\mathbf{x}$ can be decomposed as follows

$$E_{LS}\{E_{y|\mathbf{x}}\{(y - \hat{y}(\mathbf{x}; LS, k))^2\}\} = \sigma^2 + \left[f(\mathbf{x}) - \frac{1}{k}\sum_{l=1}^{k} f(\mathbf{x}_{(l)})\right]^2 + \frac{\sigma^2}{k} \tag{1}$$

   where $\hat{y}(\mathbf{x}; LS, k)$ is the prediction of the kNN method for a point $\mathbf{x}$ given a learning sample LS (of size N), $x_{(l)}$ denotes the l^{th} nearest neighbours of $\mathbf{x}$ in LS and k is the number of neighbours.

2. Let us now assume that the problem is unidimensional, i.e., $x \in \mathcal{R}$, that $f(x) = x^2$, and that for a given N, the inputs of the training examples are defined as follows:

$$\{x^1, \ldots, x^N\} = \{0\} \cup \{\frac{i}{N'} | i = 1, \ldots, N'\} \cup \{-\frac{i}{N'} | i = 1, \ldots, N'\}, \tag{2}$$

   with $N' > 1$ an arbitrary integer value. The N $(= 2N' + 1)$ points thus form an uniform grid in $[-1; +1]$. Using the result in (1), express analytically the bias and variance terms of the kNN method at $x = 0$, as a function of k, N and σ. You can assume that k only takes the values $k = 2k' + 1$ with $k' \geq 0$ an integer value chosen in $\{0, 1, \ldots, N'\}$.

3. Discuss the impact of N, k, an σ on bias and variance. Are there some surprising or missing dependences? If so, try and explain them.

4. For all combinations of N in $\{25, 50\}$ and $\sigma \in \{0.0, 0.1, 0.2\}$, determine the value k^* of k that minimizes the expected generalization error at $x = 0$.[2]

5. Discuss the impact of N and σ on k^*.

[1] `https://www.gradescope.com`, Entry code: EVGWRX.

[2] We expect numerical values here, not an analytical solution. You can either obtain them by finding the minimum of the bias plus variance term found in 2, or by running actual experiments.

2 Empirical analysis

In this section, we assume that we have access to a (large) dataset (or pool) of N_S pairs $P = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_{N_S}, y_{N_S})\}$ from which we would like to design an experiment to estimate the bias and variance of several learning algorithms. We assume that N_S is large with respect to the size N of learning samples for which we want to estimate bias and variance. In the questions below, the different terms are to be understood as their mean over the input space ($E_{\mathbf{x}}$), not at a particular point $\mathbf{x}_0$ as in the previous section.

Data. For the experiments below, we propose to use the *Wine Quality* dataset available on OpenML[3]. This dataset contains $N_S = 6\,497$ samples, described by 11 input features. The goal is to predict a score between 0 and 10 for how good a wine is based on its characteristics, such as sulfites.

Note: in the following questions, we do not tell you precisely how to set all parameter values or ranges. It is your responsibility to choose them wisely to illustrate the expected behaviors. Some of the experiments may take time depending on your computer. Be patient.

(2.1) Explain why estimating the residual error term is very difficult in this setting.

(2.2) Describe a protocol to nevertheless estimate variance, the expected error, as well as the sum of the bias and the residual error from a pool P. Since the residual error is constant, this protocol is sufficient to assess how method hyper-parameters affect biases and variances.

(2.3) Implement and use this protocol on the given dataset to estimate the expected error, variance, and the sum of bias and residual error, for Lasso regression, kNN, and regression trees. For all three methods, plot the evolution of the three quantities as a function of its main complexity parameter (respectively, λ[4], k, and maximum depth) on bias and variance. You can fix the learning sample size N to 250 for this experiment. Briefly discuss the different curves with respect to the theory.

(2.4) For the same three methods, show the impact of the learning sample size on bias and variance. In the case of kNN and Lasso regression, choose one particular value of k and λ respectively. In the case of regression trees, compare fully grown trees with trees of fixed depth.

(2.5) Two so-called *ensemble* methods to address variance and bias are bagging ("bootstrap aggregating") and boosting. Both are based on combining weaker models to produce more accurate and stable results. Whereas bagging combines models in parallel, boosting does so in a sequential manner. Compute and discuss the evolution of bias and variance as the number of estimators[5] increases, for both bagging and boosting[6], using a decision tree as your base learner. Discuss as well how the complexity of the base learner (e.g., the maximum depth) affects the ensembling results.

[3] A python script is provided on the project website to retrieve it.
[4] It is denoted λ in the lecture slides. In scikit-learn, this parameter is denoted α.
[5] `n_estimators` in Scikit-Learn
[6] You can use the implementations of `AdaBoostRegressor` and `BaggingRegressor` in scikit-learn.