Assignment 1 - Trade-off between Overfitting and Underfitting

Due Date: **September 27, 2025, 11:59 pm**
Assessment: 5% of the total course mark.

Description:

In this assignment you are required to perform an experiment similar to the experiment in Section 1.2 of Bishop[1]. **It is strongly recommended that you read carefully Section 1.2 of [1].**

You will have to train and compare several regression models to illustrate the trade-off between overfitting and underfitting. You will use data generated synthetically. The prediction is to be performed based on only one feature, denoted by x. The target t is a noisy measurement of the function $f_{true}(x) = sin(2\pi x)$. Thus, t satisfies the following relation

$$t = sin(2\pi x) + \epsilon \tag{1}$$

where ϵ is random noise with a Gaussian distribution with 0 mean and variance 0.04.

Note that for this scenario, we know the probability distribution of the unseen data (x, t). Therefore, we can compute the optimal predictor $f_{opt}(x)$ (i.e., the predictor that achieves the **smallest expected squared error**). Namely,

$$f_{opt}(x) = sin(2\pi x). \tag{2}$$

In this experiment, you will only have a small training data set and you will design several predictors by training several models on this training data. Then you will compare the predictors obtained with the optimal predictor $f_{opt}(x)$.

Construct a training set consisting of only $N = 9$ data samples $\{(x^{(1)}, t^{(1)}), \cdots, (x^{(N)}, t^{(N)})\}$, where $x^{(1)}, \cdots, x^{(N)}$ are uniformly spaced in the interval $[0, 1]$, with $x^{(1)} = 0$, $x^{(N)} = 1$, and $t^{(1)}, \cdots, t^{(N)}$ are generated using relation (1). Construct a validation set consisting of 100 data samples with the feature x uniformly spaced in the interval $[0, 1]$ and targets generated randomly according to relation (1). Similarly, construct a test set with 100 data samples . You may use the code provided on the next page. **When generating the random data use a four-digit number containing the last 4 digits of your student ID (in any order), as the seed for the pseudo number generator.**

First you have to train $N - 1$ regression models of increasing capacity, namely

$$f_M(x) = w_0 + w_1 x + w_2 x^2 + \cdots + w_M x^M, \tag{3}$$

where $1 \le M \le N - 1$. For each M, you have to train the model by minimizing the average squared error on the training set and record the training and validation **root mean squared errors** (RMSE). To train the model given by equation (3), you can think of it as a linear regression model, where the feature vector $\mathbf{x}$ has dimension M and is equal to $(x, x^2, x^3, \cdots, x^M)^T$. Then just apply the formula given in class to obtain the vector of parameters $\mathbf{w}$ that minimizes the training error.

For each M, plot the predictor function $f_M(x)$ and the curve $f_{opt}(x)$ for $x \in [0, 1]$. Additionally, include in the figure all the points in the training set and in the validation set with their true target values. Use different colours for the training examples, the validation examples, the trained prediction function and the optimal predictor. Have separate figures for different values of M. In a separate figure, plot the training and validation RMSEs versus M, for all $N-1$ predictor functions that you obtained. Also include in this plot the RMSE on the validation set corresponding to the optimal predictor f_{opt} (what do you notice about it?). In your report, identify the regions (i.e., values of M) where overfitting and underfitting occur, respectively and explain your choice. Also identify the optimal capacity region.

Additionally, for $M = N-1$ you have to train the model with L^2 regularization in order to mitigate overfitting. Here you have to consider more values for the hyperparameter λ, namely $\lambda = 10^i$, for all integeers i between -14 and 2 inclusive. Then plot the training and validation RMSEs versus $\log_{10} \lambda$. In your report, specify the regions (i.e., values of λ) where overfitting and underfitting occur, respectively and explain your choice. Also identify the optimal capacity region. Next, pick three values of λ, namely, $\lambda_1, \lambda_2, \lambda_3$, such that λ_1 is in the overfitting region, λ_2 is in the optimal capacity region, and λ_3 is in the underfitting region. For each of these three values of λ, plot the prediction function obtained by training with regularization and the curve $f_{true}(x)$ for $x \in [0, 1]$. Also include all the points in the training set and in the validation set with their true targets. Have separate figures for different values of λ.

Among all models that you have trained select the best model and explain your choice. Then measure its performance (i.e., the RMSE) on the test set and specify it in the report.

You have to write a report to present your results and their discussion. The report should contain all the plots and explanations mentioned above. Also report the parameter vectors for λ_1, λ_2 and λ_3 Do they fulfill your expectations? Are the observations made from the plots of the errors (for all the trained models) consistent with the observations made from the plots of the curves of the prediction functions? How close are the trained predictors to the optimal predictor f_{opt}? The report should be clear and concise.

Beside the report, you have to submit your numpy code used for the experiment. The code has to be modular. Write a function for each of the main tasks. Also, write a function for each task that is executed multiple times (e.g, to compute the average error). **Use vectorization, where possible!. The code should include instructive comments. You have to write the code for training your models yourself. You are allowed to use from sklearn only functions to standardize the features.**

Code for data generation:

You may use the following code for generating the training and the validation sets:

```
import numpy as np
X_train = np.linspace(0.,1.,N) # training set
X_valid = np.linspace(0.,1.,100) # validation set
```

```
np.random.seed(...)
t_valid = np.sin(2*np.pi*X_valid) + 0.2 * np.random.randn(100)
t_train = np.sin(2*np.pi*X_train) + 0.2 * np.random.randn(10)
```

Code for feature standardization:

When training a linear model with regularization, the features have to be standardized first. You may use the code given below for feature standardization.

```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
XX_train = sc.fit_transform(XX_train)
XX_valid = sc.transform(XX_valid)
```

Note that the standardization is performed based on the training data. Then the same transformations are applied to the validation data in order to be able to use the trained predictor on the validation data. In other words, the value that is subtracted from each feature and the value used for scaling in the validation data are the same ones that were used for the training data.

Submission Instructions:

- Submit the report in pdf format and the python file (with extension ".py") containing your code. Submit the files in the Assignment's Box on Avenue.

References

[1] C. M. Bishop, Deep Learning: Foundations and Concepts, Springer, 2024 (ISBN 9783031454684), a free-to-use online version is available at `https://www.bishopbook.com/`