McMaster University
Dept. of Electrical and Computer Engineering

COE 4SL4 - Term I (Fall) 2024

Assignment 1 - Trade-off between Overfitting and Underfitting

Due Date: **September 28, 2024, 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 1.1 of PRML [1]. **It is recommended that you read carefully Section 1.1 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 $f_{true}(x) = sin(4\pi x)$. Thus, t satisfies the following relation

$$t = \underbrace{sin(4\pi x)}_{f_{true}(x)} + \epsilon \tag{1}$$

where ϵ is random noise with a Gaussian distribution with 0 mean and variance 0.09. Note that $f_{true}(x)$ is the optimal predictor for the data generated with equation (1) (i.e., the predictor that achieves the smallest expected squared error). However, in this experiment, you will only have a small training data set and you will design the predictor only based on the knowledge of the training set.

Construct a training set consisting of only 10 examples $\{(x^{(1)}, t^{(1)}), \cdots, (x^{(10)}, t^{(10)})\}$, where $x^{(1)}, \cdots, x^{(10)}$ are uniformly spaced in the interval $[0, 1]$, with $x^{(1)} = 0$, $x^{(10)} = 1$, and $t^{(1)}, \cdots, t^{(10)}$ are generated using relation (1). Construct a validation set consisting of 100 examples 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 examples. 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 ten regression models of increasing capacity (corresponding to M from 0 to 9) and record and compare their training and validation errors. For each M, $0 \leq M \leq 9$, the predictor has the form

$$f_M(x) = w_0 + w_1 x + w_2 x^2 + \cdots + w_M x^M.$$

Note that, when $M = 0$, the predictor is just a constant function. For each M, you have to train the model using the squared difference as the loss function and record the training and validation **root mean squared errors**. For each M, plot the predictor function $f_M(x)$ and the curve $f_{true}(x)$ versus $x \in [0, 1]$. 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 prediction function and the true function. Moreover, plot the training and validation **root mean squared errors** versus M, for all twelve predictor functions that you obtained. Also include in this plot the average **root mean squared errors** between the targets and the true function $f_{true}(x)$ for the examples in the validation set. In your report, explain what the overfitting, underfitting and the optimal capacity regions are (i.e. the values of M) and explain how you chose them.

Additionally, for $M = 9$ you have to train the model with regularization in order to control overfitting. Here you have to try more values for λ until you find an "optimal" value λ_1 that eliminates overfitting. For this, choose a range for $\ln \lambda$ and increase $\ln \lambda$ with constant size increments. This range has to be appropriately chosen to cover the underfitting, overfitting and optimal capacity scenarios. Note that in [1], an appropriate range is approximately from -40 to 0. In this assignment, the parameters of the data are slightly different, so the results will not be the same as in [1], but you could expect a similar range for the $\ln \lambda$ values to be appropriate. Then plot the training and validation errors versus $\ln \lambda$. In your report, explain what the overfitting, underfitting and the optimal capacity regions are (i.e. the ranges of λ) and explain how you chose them.

Identify a value λ_1 that eliminates overfitting. Identify a value λ_2 for which underfitting occurs. For each of λ_1 and λ_2 plot the prediction function $f_M(x)$ and the curve $f_{true}(x)$ against x, and all the points in the training set and in the validation set with their true targets.

Among all models that you have trained select the best model. Then measure its performance on the test set.

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 include in this plot the average **root mean squared errors** between the targets and the true function $f_{true}(x)$ for the examples in the validation set. Justify your choice for λ_1 and λ_2. Report the parameter vectors for λ_1 and λ_2 and compare them. 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? 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). **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.,10) # training set
X_valid = np.linspace(0.,1.,100) # validation set
np.random.seed(...)
```

```
t_valid = np.sin(4*np.pi*X_valid) + 0.3 * np.random.randn(100)
t_train = np.sin(4*np.pi*X_train) + 0.3 * 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, the python file (with extension ".py") containing your code, and a short demo video. The video should be 2 min or less. In the video, you should scroll down your code and explain what each part does. Submit the files in the Assignments Box on Avenue.

References

[1] C. M. Bishop, Pattern Recognition and Machine Learning, Springer, 2006 (ISBN 9780387310732), available for free download at https://www.microsoft.com/en-us/research/people/cmbishop/prml-book/.