Machine Learning
Università della Svizzera italiana

Assignment 1

Morales Mariciano Jeferson

May 2, 2024

!! !! Note that this file is just meant as a template for the report, in which we reported *part of* the assignment text for convenience. You must always refer to the text in the README.md file as the assignment requirements !! !!.

Tasks

This section should contain a detailed description of how you solved the assignment, including all required statistical analyses of the models' performance and a comparison between the linear regression and the model of your choice. Limit the assignment to 8-10 pages and do not include any code in the report.

Task 1

Use the family of models $f(\mathbf{x}, \boldsymbol{\theta}) = \theta_0 + \theta_1 \cdot x_1 + \theta_2 \cdot x_2 + \theta_3 \cdot \cos(x_1) + \theta_4 \cdot x_2 \cdot x_2 + \theta_5 \cdot \tanh(x_1)$ to fit the data.

a. Write in the report the formula of the model substituting parameters $\theta_0, \ldots, \theta_5$ with the estimates you've found:

$$f(\mathbf{x}, \boldsymbol{\theta}) = \_ + \_ \cdot x_1 + \_ \cdot x_2 + \_ \cdot \cos(x_1) + \_ \cdot x_2 \cdot x_2 + \_ \cdot \tanh(x_1)$$

b. Evaluate the test performance of your model using the mean squared error as performance measure.

c. Implement Lasso Regression, what do you observe? What can you infer about the given family of models?

Task 2

Consider any family of non-linear models of your choice to address the above regression problem.

a. Evaluate the test performance of your model using the mean squared error as performance measure (same data as Task 1).

b. Compare your model with the linear regression of Task 1. Which one is statistically better?

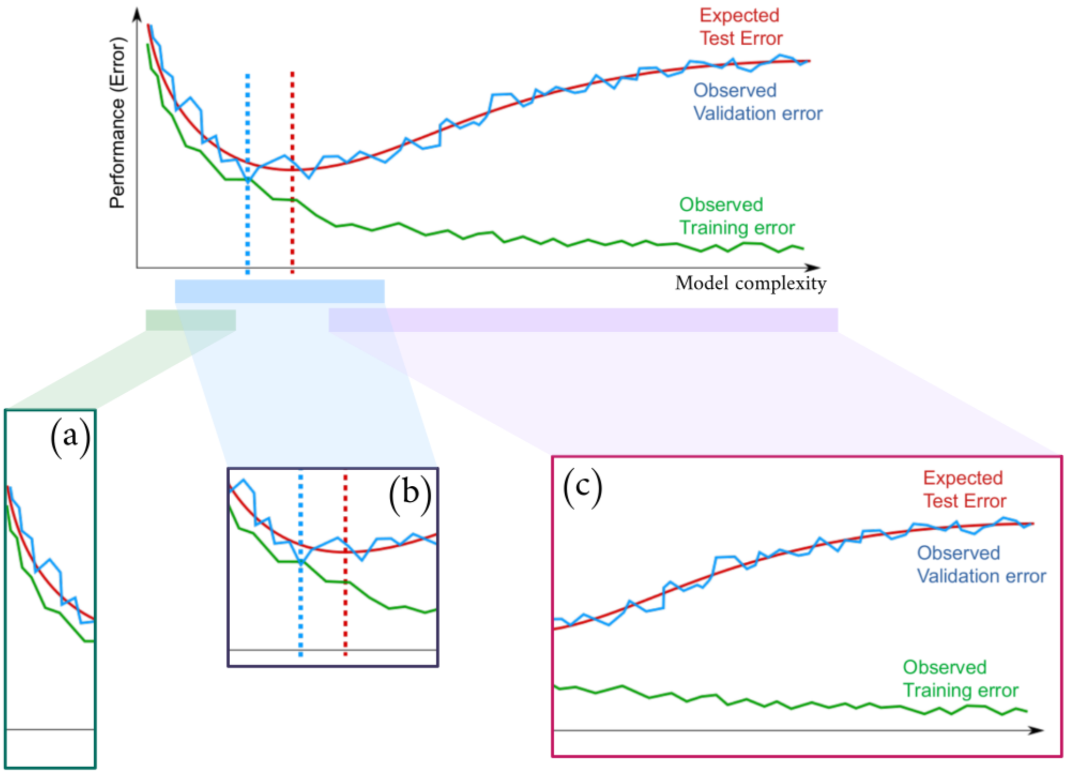
Task 3 (Bonus)

In the **GitHub repository of the course**, you will find a trained Torch learn model that we built using the same dataset you are given (**data_bonus**). This **baseline** model is able to achieve a MSE of **0.013**, when evaluated on the test set. You will get extra points if you provide a model of your choice whose test performance is **better** (i.e., the MSE is lower) than ours. Of course, you must also tell us why your model is performing better.

Questions

Q1. Training versus Validation

Figure 1: Training versus validation exercise image



Q1.1 What is the whole figure about?

A1.1

The Figure 1 shows the bias-variance tradeoff for a model within the context of the regression problem. The bias gives errors due to erroneous assumptions in the learning procedure. In contrast, the variance yields errors due to the sensitivity of the model to the training set fluctuations i.e. noise in the training dataset. The three error lines displayed in the plot correspond to:

- Observed training error, from training set used to fit the model on training data
- Observed validation error, from test set assessing performance meaning how well the model generalizes to unseen data
- Expected test error, from an ideal unbiased dataset assessing performance on unseen data meaning how well the model would ideally generalize to unseen data

Along the x-axis, model complexity is intended as more hidden layers or neurons in the model. In this particular model we can see that the observed training error is decreasing as the model complexity increases, this will lead to overfitting. Along the y-axis, the lower the performance metric, the better the model since it measures the error of the model.

Q1.2 Explain the behaviours of the curves in each of the three highlighted sections in the figure, namely (a), (b), and (c).

A1.2

Q1.2.a Can you identify any signs of overfitting or underfitting in the plot? If yes, explain which sections correspond to which concept.

A1.2.a

Q1.2.b How can you determine the optimal complexity of the model based on the given plot?

A1.2.b

Q1.3 Is there any evidence of high approximation risk? Why? If yes, in which of the below subfigures?

A1.3

Q1.4 Do you think that increasing the model complexity can bring the training error to zero? And the structural risk?

A1.4

Q1.5 If the X axis represented the training iterations instead, would you think that the training procedure that generated the figure used early stopping? Explaib why. (**NB:** ignore the subfigures and the dashed vertical lines)

A1.5

Q2. Linear Regression

Comment and compare how the (a.) training error, (b.) test error and (c.) coefficients would change in the following cases:

Q2.1 $x_3 = x_1 + 0.2 \cdot x_2$.

A2.1

Q2.2 $x_3 = x_1 * *2$ (in Python ** is the "power" operator, so 3 ** 2 = 3 * 3 = 9).

A2.2

Q2.3 x_3 is a random variable independent from y.

A2.3

Q2.3 How would your answers change if you were using Lasso Regression?

A2.3

Q2.4 Explain the motivation behind Ridge and Lasso regression and their principal differences.

A2.4

Q3. Logistic Regression

Q3.1 What are the main differences between the logistic-regression and the perceptron?

A3.1

Q3.2 Discuss the major limit they share and how neural networks can solve it.

A3.2

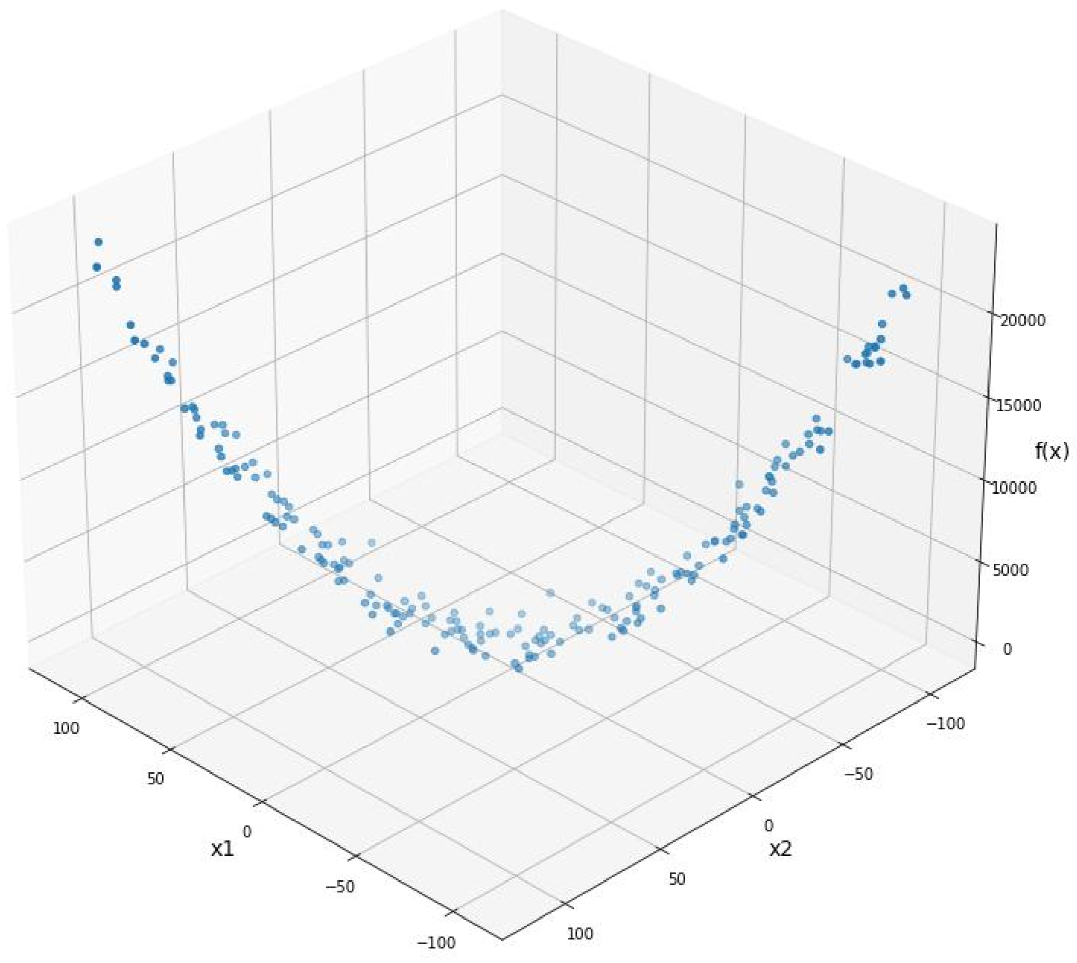
Q3.3 What is the role of activation functions in feedforward neural networks.

A3.3

Q4. Consider the regression problem shown in the picture 2 below and answer each point.

Q4.1 Do you think a model of the family f(x, theta) = $\theta_0 + \theta_1 * x_1 + \theta_2 * x_2$ is a good choice for such task? Why?

Figure 2: Regression problem



A4.1 The simple linear model $f(x, \theta)$ would not be a good fit for the data. It is evident that the data does not follow a linear trend. The linear model would lead to underfitting where the model is not able to capture the underlying patterns. The relationship between the inputs variables and the dependent output variable seems to be non-linear, describing a quadratic interaction on both x_1, x_2 resulting in a parabolic shape. A linear relationship among input variables as in $f(x, \theta)$ would not capture the curvature form of the data.

Q4.2 Do you think using a feed-forward neural network would improve the results?

A4.2 Using an FFNN would be a good choice for this problem task. It is for sure better than the simple model with linear relationship between independent input variable x_1, x_2 and dependent output variable $f(x, \theta)$. In general, with FFNNs we can leverage the *Universal Approximation Theorem (1991)* which states that an FFNN with a single hidden layer containing a finite number of neurons and a linear ouput neuron can approximate any continuous function on compact subsets of $\mathbb{R}^n$. Moreover, FFNN are highly flexible and can approximate virtually any function, so complex non-linear relationship patterns, which are hard for linear and polynomial regression, are handled through the encapsulated abstraction of the neurons in the hidden layer(s). In this case, the parabolic shape of the data would be captured, as FFNN are a common choice for regression tasks with non-linear data.