Machine Learning (BSc - SP24)
Università della Svizzera italiana

# Assignment - 1

Munkhjargal Munkhgerel

May 3rd, 2024

## Tasks

### 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:

   $f(\mathbf{x}, \boldsymbol{\theta}) = 0.969346 + 5.0888167 \cdot x_1 + -4.003447 \cdot x_2 + 7.0922946 \cdot \cos(x_1) + 1.9960 \cdot x_2 \cdot x_2 + -0.1479577 \cdot \tanh(x_1)$

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

   – Linear Regression - Train performance (MSE): **1.42**

   – Linear Regression - Test performance (MSE): **1.45**

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

   – Lasso Regression - Train performance (MSE): **6.936**

   – Lasso Regression - Test performance (MSE): **6.367**

- *Model Performance:* The Linear Regression model outperforms the Lasso Regression model in terms of MSE, both on the Train and Test sets. This suggests that the LR model is more accurate in predicting the target variable.
- *Overfitting*: Lasso Regression performs worse on the Test set (6.367 vs 1.450) and the Train set (6.935 vs 1.419). This indicates that the Lasso Regression model is overfitting the Train data, meaning it became too specialized and did not generalize well to new, unseen data. In contrast, the LR model is less prone to overfitting.

- However, I also used the cross-validation which allows it to evaluate the performance of the estimator for different sets of hyperparameters. MSE results were same as Linear Regression results.

  - Alternative Lasso Regression - Train performance (MSE): **1.4197**
  - Alternative Lasso Regression - Test performance (MSE): **1.4500**

  - Linear Regression with Gradient Descent - Train (MSE): **1.816**
  - Linear Regression with Gradient Descent - Test (MSE): **1.837**

### 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).
   - Polynom Regression - Train performance (MSE): **1.399**
   - Polynom Regression - Test performance (MSE): **1.47**

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

- These results suggest that Polynomial Regression and Linear Regression are similarly effective, while Lasso Regression struggles with both overfitting and underfitting.
- Model selection: Based on the results, Polynomial Regression and Linear Regression appear to be top contenders for model selection.
- Model complexity: Polynomial Regression is a more complex model compared to Linear Regression. However, the results indicated that the added complexity doesn't necessarily improve the model's ability to generalize to new data.
  - Paired t-test results:
  - T-statistic: 0.028123277867955196
  - P-value: 0.9821008831663618
  - Fail to reject the null hypothesis: There is no significant difference in performance between linear regression and polynomial regression.

## Questions

### Q1. Training versus Validation

Q1.1 What is the whole figure about?

A1.1 In this plot, it shows the relationship between model complexity and performance. The plot shows training error and test error as a function of model complexity. When the model is less complex than the optimal, we underfit (in the left side of the plot) the data. This means the model is too simple to capture the underlying patterns in the data (or we do not consider the structure of the data), resulting in poor performance both on the training set and the test set. Conversely, when the model complexity exceeds the optimal point (right side of the plot), we overfit the data which means that we focus on the noise of the data as opposed to the underlying structure. In this overfitting side, the model is too flexible and captures noise in the training data, leading to excellent performance on the training set but poor generalization to unseen data or high-test error. To get the best results, it's important to find the right amount of complexity in a model. This balance, known as the bias-variance trade-off, shows that if a model is too simple, it might not capture all the important patterns in the data (underfitting). On the other hand, if it's too complex, it might fit the training data too closely and struggle with new data (overfitting). The goal is to find the perfect level of complexity where the model can understand the data well without fitting it too tightly or too loosely.

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 In picture a, expected performance is expected. We average up all the possible runs that we might consider, otherwise it would be binary - In b, the horizontal blue dashed line indicates the optimal model according to the model selection or validation set, whereas the red dashed line indicates the optimal point. In between these two points is the ideal range (best fit) for model complexity

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 Overfitting increases the gap between the observed validation error/expected test error and observed training error (from horizontal red dashed line, the optimal model,to the right side of the picture) when the model complexity increases. Conversely, it indicates underfitting when both training and test errors are high or similar while the model complexity decreases.

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

A1.2.b To determine the optimal complexity of the model, we need to observe the how both training and testing errors change with varying model complexity. The points:

- When the model's complexity is low, both training and testing errors are high,
- With increasing complexity, the training error decreases. However, the testing error initially decreases until a certain point, after which it starts to increase.

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

A1.3 Yes, the high approximation risk can be found on subfigure C where there is a huge gap between the observed validation error/expected test error and observed training error (overfitting area) as the model complexity increases. (This gap indicates that the model is unable to generalize well to unseen data despite performing well on the training set)

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

A1.4 When we increase the model complexity, the training error decreases. So, it brings the training error closer to zero which indicates the overfitting where the model has memorized the training data but fails to generalize well to unseen data. Thus, overfitting leads to poor performance on new, unseen data, resulting in a higher generalization error or structural risk.

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? Explain why. (**NB:** ignore the subfigures and the dashed vertical lines)

A1.5 Early stopping is beneficial because it helps to prevent the model from continuing to learn and adapt to the training data beyond a certain point. As training iterations progress, the model's performance on the training data typically improves, leading to a decrease in the training error. However, at some point, further training iterations may start to cause the model to overfit the training data, resulting in an increase in the generalization error or structural risk.

## 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 $x_3 = x_1 + 0.2 \cdot x_2$.

| | Training error | Test error | Coefficients |
|---|---|---|---|
| A2.1 | When we add the new regressor $x_3$, the model becomes more flexible and complex, thus the training error might decrease, which indicates a better fit to the training data. | The test error might decrease, indicating that the model is generalizing better to new, unseen data. However, if the new regressor $x_3$ adds noise or overfits to the training data, the test error might increase. | The coefficients $\theta_0$, $\theta_1$, and $\theta_2$ might change to accommodate the new feature $x_3$, and $\theta_3$ would represent the weight of $x_3$ in the model. |

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

| | Training error | Test error | Coefficients |
|---|---|---|---|
| A2.2 | This new regressor captures non-linear relationships between $x_1$ and the target variable $y$, allowing the model to better fit the training data. As a result, the training error would decrease. | The non-linear relationship captured by the new regressor $x_3$ might improve the model's generalization to new, unseen data. However, if the relationship is overly complex or contains noise, the test error might not necessarily decrease. | The coefficients of the original regressors $x_1$ and $x_2$ ($\theta_1$ and $\theta_2$) might change significantly due to this new added non-linearity regressor. The coefficient for $x_3$ ($\theta_3$) will likely be significant, indicating the importance of this non-linear term. |

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

A2.3

| Training error | Test error | Coefficients |
|---|---|---|
| The relationship between $x_3$ and $y$ is independent, which means the model is not leveraging any new insights or patterns in the data. Therefore, the training error might not change significantly or could increase if the added regressor is poorly correlated with the target variable $y$. | Since $x_3$ is independent from $y$, the new regressor does not provide any useful information for predicting the target variable $y$. As a result, the test error might not decrease or could increase due to the added noise or complexity introduced by the new regressor. | The coefficients of the original regressors $x_1$ and $x_2$ ($\theta_1$ and $\theta_2$) might remain unchanged, as the new regressor $x_3$ is independent of the target variable $y$. The coefficient for $x_3$ ($\theta_3$) will likely be non-significant or close to zero, indicating that this new regressor does not contribute to the prediction task. The intercept $\theta_0$ might not change significantly as well. |

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

A2.3

| Training error | Test error | Coefficients |
|---|---|---|
| Using Lasso Regression, the addition of the new regressor $x_3$ might reduce the training error, as Lasso Regression is a regularization method that adds a penalty term to the cost function to prevent overfitting. The Lasso penalty would encourage some of the coefficients to shrink towards zero which could lead to a better fit of the training data. | In Lasso Regression, the new regressor $x_3$ might not improve the test error, as the penalty term would focus on reducing the magnitude of the coefficients, especially for the new regressor. This regularization step would help prevent overfitting, which could lead to better generalization performance on new, unseen data. | Some of the coefficients might be shrunk towards zero due to the Lasso penalty. This would occur when the magnitude of the absolute value of the coefficient is greater than the Lasso regularization alpha. As a result, some regressors might be eliminated from the model, reducing the overall complexity of the model and improving generalization. |

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

A2.4 They are forms of regularized linear regression. When fitting a linear model to data, the goal is to minimize the mean squared error between the model and the data. The MSE quantifies how well the model fits the data, the smaller the MSE, the better the model. However, a common issue is overfitting, where the model fits the training data well which could result in a small MSE but performs poorly on new, unseen data which could lead to a high MSE. To address this, regularization techniques are used. A way of controlling overfitting is by adding an extra term to the MSE that penalizes large coefficients in the model. Ridge regularization penalizes the sum of squared coefficients which brings the coefficients closer to zero, not at zero. This helps reduce variance in the model. While Lasso regularization penalizes the sum of absolute values of coefficients which brings the coefficients exactly to zero, effectively removing the them from the model. By fine-tuning the regularization parameters, regularized linear regression models can prevent overfitting, ensuring that the MSE for both the training and test sets remains similar.

## Q3. Logistic Regression

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

A3.1 Logistic regression can be described as a stronger version of the perceptron because logistic regression can guess probabilities, while the perceptron can only say yes or no. In other words, logistic regression is linear classifier that estimates the probability of a binary outcome directly. Conversely, the perceptron is a single-layer neural network with a set of input features. It calculates a weighted sum of the input features and applies an activation function to make a binary decision based on whether the sum is above or below a certain threshold. Another difference between perceptrons and logistic regression is that perceptrons are only able to learn linear decision boundaries, while logistic regression can learn non-linear decision boundaries. This is because the logistic function is a non-linear function. Perceptron is sensitive to outliers and noisy data due to its update rule, whereas the logistic regression is more robust to outliers and noise due to probabilistic output.

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

A3.2 Logistic regression and perceptrons struggle with complex, non-linear data relationships as they rely on linear decision boundaries. Neural networks address this by employing multiple layers, including hidden layers, to learn hierarchical data structures. These layers use non-linear transformations, facilitated by activation functions such as sigmoid and ReLU, to capture intricate patterns. Additionally, techniques such as backpropagation refine neuron connections, enabling neural networks to adapt and enhance performance. Overall, neural networks outperform logistic regression and perceptrons by leveraging multi-layered architecture and non-linear functions, thereby improving their ability to model complex data relationships and enhance predictive capabilities.

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

A3.3 The role of activation functions in feedforward neural networks enables the network to understand model complex, non-linear relationships between the inputs and outputs. Without these functions, the neural network is to be only able to model linear relationships which restricts its capacity to learn and make generalizations from the data.

## Q4. Consider the regression problem shown in the picture 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?

A4.1 The given model of family $f(x, \theta) = \theta_0 + \theta_1 \cdot x_1 + \theta_2 \cdot x_2$ is a linear regression with multiple variables. In this case, the quadratic term seems missing in the linear model, which means that the model would not be able to capture the curvature. This would lead to a poor fit of the training data and a high predicted error on new, unseen data. In this case, it is suitable to use a polynomial regression model. The polynomial can have as many custom features as we wish to fit the data. Polynomial regression considers higher-order terms of the input variables, allowing us to capture non-linear patterns in the data.

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

A4.2 Using a feedforward neural network could improve the results compared to the linear regression model. Neural networks can learn complex between the inputs and the target variable by incorporating multiple hidden layers and non-linear activation functions.

## References

- Lecures, Lab sessions, Youtube videos
- https://machinelearning101.readthedocs.io/en/latest/
- https://www.svms.org/srm/
- https://mlweb.loria.fr/book/en/contents.html
- https://medium.com/@rakeshandugala/how-artificial-neural-networks- -from-perceptrons-to-gradient-descent-28c5552d5426
- http://christiansch.github.io/machine-learning-cheat-sheet/
- https://bhatnagar91.medium.com/how-neural-networks-learn-using-grad {ient-descent-f48c2e4079a6
- https://www.geeksforgeeks.org/machine-learning/?ref=lbp
- https://machinelearningmastery.com
- https://datascience.stackexchange.com/
- https://dev.to/t/machinelearning