Data-Driven Design & Analysis of Structures & Materials (3dasm)

Miguel A. Bessa (M.A.Bessa@tudelft.nl)

Delft University of Technology, The Netherlands

Problem statement for Midterm Project

This assignment intends to make you reflect on what you have learned so far about the basics of machine learning. Each group has to deliver **one PDF report** of this assignment containing the answers to the questions listed here, and the corresponding **Jupyter Notebooks** used to answer the questions.

Due date: March 25, 2022

1 General instructions

- Your **PDF report** should be divided in 3 parts, as done in this problem statement. Each section/subsection of your report should clearly indicate the question/subquestion that is being answered.
- Deliver all your files in a single **ZIP folder** named "3dasm_Midterm_Group$X$" where X is your Group number. This folder should be organized as follows:
  - Include the report as a PDF named "3dasm_Midterm_Report_Group$X$.pdf"
  - Include **three** Jupyter Notebooks. One for each Part of this assignment. Each notebook is named "3dasm_Midterm_Part$N$_solution.ipynb" where N is the Part number (1, 2 or 3).
  - Include a folder called "docs" where you can include additional files needed to run the notebooks (for example, Pandas dataframes).

2 Part 1: Linear regression models in 1D

Consider the function to be learned as $f(x) = x\sin(x)$ within the domain $x \in [0, 10]$.

1. Conceptual questions.
   - 1.1. Explain from a Bayesian perspective the Linear Least Squares model and why do we use it to fit data with polynomial basis functions. Include a derivation for the point estimate of the parameters and the Posterior Predictive Distribution (PPD).
   - 1.2. Explain what are training, validation and testing sets.
   - 1.3. Define and explain what are the R^2 and MSE metrics.
   - 1.4. Explain what is k-fold cross-validation.
   - 1.5. Define what are hyperparameters and parameters in machine learning models.
2. Linear regression model for dataset without noise.
   - 2.1. Create a dataset without noise from the $f(x) = x\sin(x)$ function. **Differently from what was done in class**, create the dataset as follows:
     - Randomly sample 40 points from $f(x)$ within the defined domain ($x \in [0, 10]$).
     - After sampling the points, sort them in ascending order (from smaller to larger x).
     - Save the dataset as a pandas dataframe and create (in the "docs" folder) a corresponding file called "Noiseless_dataset.csv".
     - Split the dataset in two sets (training and testing sets) using the "train_test_split" function of scikit-learn and consider 80% of the data is included in the training set. Set the "random_state" seed to a particular value and **mention in the report the value of the seed**.
     - **Important**: make sure that you use the same training and testing sets for fitting all models for this question as well as the remaining ones.

2.2. Calculate the R^2 and MSE on the testing set after training Linear Regression models with polynomials of different degree (1, 3, 5, 10, 20) using three different training set sizes (respectively with the first 6, 11 and 21 points of the training set defined in 2.1). Present the previously mentioned error metrics as suggested in Table 1 and show 3 figures with the plots considering 6, 11 and 21 training points (one figure per row of the table). **What can you conclude from these results**?

Table 1: Suggested table to report R^2 (similar for MSE) in each cell.

degree / n_{train}	1	3	5	10	20
6					
11					
21					

2.3. Now consider the **entire training set** defined in 2.1 and use 6-fold cross validation to calculate the **mean** and **standard deviation** of the R^2 and MSE metrics when training Linear Regression models with polynomials of different degree (1, 5, 20). Present the result as suggested in Table 2. **Explain the obtained mean and standard deviation for the error metrics**.

Table 2: Suggested table to report R^2 and MSE using cross-validation where each cell contains their mean and standard deviation.

degree / n_{train}	1	5	20
MSE	mean $\pm$ std	mean $\pm$ std	mean $\pm$ std
R^2	mean $\pm$ std	mean $\pm$ std	mean $\pm$ std

3. Linear regression model for dataset **with** noise.

3.1. From the dataset you created in 2.1, perturb the output values of each point **with** noise. Consider the same type of noise used in Lecture 9[1].

- Once you generated the new dataset (same x values but perturbed y values), save it as a pandas dataframe and create (in the "docs" folder) a corresponding file called "Noisy_dataset.csv".
- As before, split the dataset in two sets (training and testing sets) using the "train_test_split" function of scikit-learn and consider 80% of the data is included in the training set. Set the "random_state" seed to **the same value you used in 2.1**.

3.2. Repeat 2.2. (but now for the noisy dataset).

3.3. Repeat 2.3.

3.4. Explain from a Bayesian perspective what is Ridge Regression and why do we use it to fit data with polynomials. Include a derivation for the point estimates of the parameters and the Posterior Predictive Distribution (PPD).

3.5. Repeat 2.2. but now using Ridge Regression with hyperparameter $\alpha = 10^{-4}$. What happens when you use a bigger or smaller value of α?

3.6. (**BONUS[2] question**) Use grid search and cross-validation to determine a good alpha parameter for Ridge regression. Report the improvement of the prediction when compared to 3.5.

3.7. (**BONUS question**) Repeat 2.2 and 2.3 for the following linear regression models: Lasso, Elastic-Net and Bayesian linear regression. Include grid search and cross-validation determination of the hyperparemeters of each model.

[1]Noise was created for each point from a Gaussian distribution whose standard deviation at that point comes from a random value between 0.5 and 1.5

[2]Only do the BONUS questions if you finished all regular questions. These questions award additional points (up to the top mark of 100%)

3 Part 2: Gaussian processes and Neural Networks in 1D

3. Gaussian Process Regression (GPR).

   3.1. In the context of Gaussian Processes, explain in your own words (1) what is a kernel function, (2) what are the parameters and hyperparameters of this model; and (3) what is the role of adding a noise term to the diagonal of the covariance matrix.

   3.2. Repeat 2.2 (noiseless dataset) using Gaussian Processes and where the columns of the table become the choice of different kernels: RBF, Mattern 5-2, Mattern 3-2, and Exponential-Sine-Squared. Report the final parameters that result from the optimization.

   3.3. Repeat the above question but now for the noisy dataset and using Gaussian Processes.

4. Artificial Neural Networks (ANN).

   4.1. Explain what are neurons (in ANN), weights, biases, epochs, batch size and gradient descent optimization.

   4.2. Repeat 2.2. (noiseless dataset) considering an ANN with **one hidden layer only** and where the columns of the table become the number of neurons in that layer: 1, 4, 16, 64, 256. Use the following hyperparameters for the network training: batch size equal to training set, 1000 epochs, early stopping with patience 30 for the minimum validation loss, ReLu activation function and ADAM optimizer.

   4.3. Repeat the above question for the noisy dataset.

   4.4. One strategy to avoid overfitting in ANNs is called "early stopping", as you considered in the previous questions. However, given the complexity of neural networks, early stopping can also have a negative effect on training. Consider the noiseless dataset use 5 training points (first 5 of the training set) to train a network with two hidden layers with the following hyperparameters: 200 neurons for first hidden layer, 10 neurons for the second hidden layer, batch size equal to training set, 1000 epochs, ReLu activation function and ADAM optimizer. Train the network for two different conditions: (1) without considering early stopping; and (2) considering early stopping with patience 30 for the minimum validation loss. Present two figures for each case (with and without early stopping) where you show the loss evolution as a function of the epochs for the training and test sets (1 figure for each case), and where you show the final approximation obtained by the networks (1 figure for each case). Probably, you have obtained a better approximation without using early stopping. What happened? Can you claim that overfitting occurred when you did not use early stopping?

   4.5. (**BONUS question**) Use grid search and cross-validation to determine better hyperparameters for the ANN. Report the improvement in the predictions when compared to 4.4.

4 Part 3: Multidimensional regression and classification

5. Multidimensional regression with GPR and ANNs.

   5.1. Create a dataset without noise as follows:

   - Select 2 noiseless test functions from each of the 6 categories of the following website: https://www.sfu.ca/~ssurjano/optimization.html. Consider the domain of analysis as suggested in the website.
   - Create a dataset similar to what you did in 2.1, but now considering 60 points. Use the same split strategy (in the same proportion of training/testing set).
   - You don't need to create a pandas dataframe for each dataset.
   - Use both methods of Part 2 (GPR and ANN) with the same hyperparameters: GPR with RBF kernel, and ANN model using the same parameters as in question 4.4 including early stopping with patience 30.

   5.2. Report the R^2 and MSE metrics in a table for the 12 functions you selected. Identify the functions that appear to be more difficult to learn and the ones that are easier and provide plausible explanations for this behavior. Please note that you will be making conclusions based on approximations that are using the same input parameters to train the algorithms, so your conclusions should not be definitive.

   5.3. Select the function with worst approximation from the previous question and try to improve the prediction for both methods using the same amount of sampling points. Report the results.

5.4. (**BONUS question**) Use an hyperparameter optimization library to find better hyperparameters for the ANN (for example: Hyperopt). Report the improvement of the prediction when compared to 5.3.

6. Multidimensional classification with Support Vector Machines and Decision Trees.

6.1. Using an adequate error metric, compare the approximation obtained when using a Support Vector Machine (SVM) with the RBF kernel and when using Decision Trees to classify the Iris dataset using 75% of points for training. You can use the same parameters we used in class, but please note that the classification task for the SVM should be performed for the 4 features (not 2). Include the six plots of the decision boundaries as a function of feature pairs obtained for both the SVM and the Decision trees.

6.2. Decision trees are prone to overfitting. A common solution is to use random forests. Explain what random forests are and why they alleviate this issue.

6.3. (**BONUS question**) Find better hyperparameters for 6.1. Report on the prediction improvements.

6.4. (**BONUS question**) Repeat 6.1 using random forests.

6.5. (**BONUS question**) Repeat 6.1 but using Gaussian Processes and ANN for this classification problem. Among the 5 algorithms you considered which one performed better? Provide a plausible explanation for this.