Practical work 04 – 14/03/2023
Universal Representation Theorem
Model Selection

Objectives

The main objectives of this Practical Work for Week 4 are the following :

a) Deepen your understanding of the Universal Representation Theorem.

b) Play through an example of overfitting and determine the optimal model complexity by using hyper-parameter tuning based on 5-fold cross validation.

Submission

- **Deadline** : Tuesday 21 March, 3pm
- **Format** :
  - Exercise 1 (Universal Approximation Theorem) :
    - pdf with your calculation (handwritten) of the gradient of the MSE cost.
    - Jupyter notebook `function_approximation_stud.ipynb` completed with your solutions. The answers to the questions can be added either in the pdf report or in the notebook.
  - Exercise 2 (Model Selection)
    - Jupyter notebook.
    - Comments and results (plot with learning curve showing the results for different model complexities) either in the notebook or in a pdf-report.

  Submission of all files in a single zip-file using the naming convention (for team of two students #1, #2) :

  `family name_given name #1- family name_given name #2.zip`

Exercise 1 Function Approximation

In this exercise, you train a single (hidden) layer neural net to represent a given function $f : [0,1] \to \mathbb{R}$. Since it is a regression problem, we will use the MSE cost function.

The MSE cost for a neural net with a 1d input x, a single hidden layer with n units and a linear output layer is given by

$$J_{\text{MSE}}(\theta) = \frac{1}{2m} \sum_{i=1}^{m} \left(y^{(i)} - \left(\sum_{k=1}^{n} w_{2,k} \sigma(w_{1,k} \cdot x^{(i)} + b_{1,k}) + b_2\right) \right)^2 \tag{1}$$

The dataset is given by suitable x-values (in the interval $[0,1]$) and associated function values $f(x)$, i.e. $\{(x^{(i)}, y^{(i)} = f(x^{(i)})) \,|\, i = 1, \ldots, m\}$.

a) Compute the formulas for gradient descent for this problem, i.e. compute the derivatives w.r.t. parameters $w_{1,k}, w_{2,k}, b_{1,k}, b_2$ and formulate the according update rules.

b) Implement MBGD for this model in the notebook `function_approximation_stud.ipynb` by completing the code where indicated by

   ### START YOUR CODE ###

   ### END YOUR CODE ###

   With the settings provided in the notebook (learning rate, batchsize, etc.) and the given dataset generated for the Beta-function (with $\alpha = \beta = 2.0$ and $m = 100$ samples) the learning should work quite well - see the learning curve (cost vs epochs) and the final MSE cost ($J_{\text{MSE}}(\theta_{\text{trained}}) \approx 7 \times 10^{-4}$).

c) Now study the impact of different settings by looking at the learning curves (as a diagnostic tool) and the cost (obtained at the end of the last epoch) as performance measure. Consider :
   - Number of epochs : How many epochs are needed to see a reasonable fit ?
   - Learning rate : How large can you choose the learning rate ?
   - Number of neurons : How many neurons are needed for a sufficiently good approximation ?
   - Batch size : What happens when you increase the batch size ?

   Can you improve the approximation as compared to the initial settings ?

d) (**Optional**) Now study different functions by generating new data with a different underlying function. Try e.g. the sine function with different frequencies. Start with a frequency $\omega = 1$ (ie. one cycle in the interval $[0,1]$). Then proceed and investigate how large the (integer) frequency can be chosen to still obtain reasonable approximations. Possibly, also consider generating a larger dataset. Give an interpretation for why the learning breaks down at larger frequencies.

Exercise 2 Model Selection

The objective of this exercise is to build a classification systems to predict whether a student gets admitted into a University or not based on their results on two exams [1].

You have historical data from previous applicants that you can use as a training set. For each training example i, you have the applicant's scores on two exams $(x_1^{(i)}, x_2^{(i)})$ and the admissions decision $y^{(i)}$. Your task is to build a classification model that estimates an applicant's probability of admission based on the scores from those two exams.

In the notebook see `overfitting_stud.ipynb`, you'll find the code to load the data and further instructions.

a) Complete the code where indicated by

   ### START YOUR CODE ###

   ### END YOUR CODE ###

   Remarks : You can use the unit tests at the end of the notebook to test your implementation of the individual methods.

b) Construct different (polynomial) models of different complexities (different degree i.e. parameter `order`). Train these models with the training set and determine the error rate on the training and the test set.

c) Determine the model best suited for the problem at hand and justify why it is the best model.

d) For all this use two different versions of the data :

   i) First version : `scores_train_1.csv` and `scores_test_1.csv` for training and testing, respectively.

   ii) Second version : `scores_train_2.csv` and `scores_train_2.csv` for training and testing, respectively.

1. Data source : Andrew Ng - Machine Learning class Stanford