Introduction to Machine Learning (IML)
BPINFOR-113
Take-Home Assignment 3 (THA3)
Grading scheme: 20 points (0-20)
Weight for final grade: 20%

Responsible: Decebal Mocanu, Christopher Gadzinski, Patrick Keller

Release date: 24 November 2023
Deadline: 22 December 2023, 23:59

Late submissions will be penalised with 2 points per week in the first two weeks after the deadline. Late submissions received after these two weeks (i.e., starting with 5 January 2024) will not be graded and will receive zero points.

Preamble

This homework has to be solved using Python. Please submit your solutions (the developed code and a report in PDF format) through Moodle. The solution needs to contain a report in the pdf format to discuss your approach and a Python file which can be easily run. State clearly in the report the group number and the group members.

The solutions for the Take-Home Assignment 3 can be submitted just by a group. Individual submissions are not allowed. Each group shall have three members.

Problem

In this homework you are requested to implement from scratch a MultiLayer Perceptron (MLP). This MLP model has to perform a two-class classification task on the dataset uploaded in Moodle. You can train the model on the data from the file "THA3train.xlsx" and validate the model on the file "THA3validate.xlsx"[1]. In the files, each row represents a data point. The first two columns (x_o, x_1) represent the input features, while the third column (y) represents the class label (0 or 1). Further on, x_o and x_1 are collected in the vector $\mathbf{x}$

The implemented MLP model has to have:

- 1 input layer ($\mathbf{x}$) with 2 neurons
- two hidden layers ($\mathbf{h^{(0)}}$, $\mathbf{h^{(1)}}$) with 10 neurons each of them.
- 1 output layer of nerons ($\mathbf{o}$) with 2 neurons (each of the two output neurons representing one class)

The relation between layers are given by:
$\mathbf{h^{(0)}} = \phi^{(0)}(\mathbf{W}_0^T\mathbf{x} + \mathbf{b}_0)$
$\mathbf{h^{(1)}} = \phi^{(1)}(\mathbf{W}_1^T\mathbf{h^{(0)}} + \mathbf{b}_1)$
$o = \phi^{(2)}(\mathbf{W}_o^T\mathbf{h^{(1)}} + \mathbf{b}_o)$

[1] Please note that we do not have and we do not use a testing dataset.

, where $\phi^{(0)}$, $\phi^{(1)}$, and $\phi^{(2)}$ represent the activation functions of that specific layer, $\mathbf{W}_0^T$, $\mathbf{W}_1^T$, $\mathbf{W}_o^T$ are weight matrices, and $\mathbf{b}_0$, $\mathbf{b}_1$, $\mathbf{b}_o$ are bias vectors.

The training has to be done using Backpropagation and Stochastic Gradient Descent.

Assignment

Please solve each of the following subproblems and give clear explanations for your solutions.

A. MLP (10 points)

1. (2 points) *Activation and Loss functions.* Please choose suitable activation functions $\phi^{(0)}$, $\phi^{(1)}$, $\phi^{(2)}$ and a suitable Loss function to perform the task (in the sense that the MLP will be able to learn and the loss function values will decrease over time). Please feel free to consider some of the activation and the loss functions presented in the lectures or to find other alternatives online. Report and justify your choices in the report.

2. (2 points) *Learning rate, batch size, initialization.* Please choose a suitable learning rate, batch size, initialization of the parameter values, and any other setting that the MLP may need in order to learn. Discuss and justify your choices in the report.

3. (2 points) *Training.* Make plots with the loss function computed over the training set and over the validation set. Stop the training process when you believe that the MLP was trained sufficiently. Justify your stopping criterium. Report the final accuracy obtained and the confusion matrix on the validation dataset.

4. (4 points) *Implementation.* We will run and check the uploaded Python file. To obtain all points for this subproblem, the Python file has to run (no errors) and the MLP model and the Backpropagation algorithm have to be implemented completely from scratch by you. You are not allowed to use any library which implements MLP models (e.g., TensorFlow, PyTorch, scikit-learn), but you are allowed to use auxiliary libraries, e.g. Numpy, Matplotlib, Pandas, and to get inspiration (no code should be copied directly as it is) from public GitHub repositories (if doing so, please acknowledge the repository in your report).

   Hint: Try to take a systematic approach and to have a logical separation of the various elements when designing your code. For instance, you can have a class implementing an MLP, and containing several methods such as forwardPass, backwardPass, weightUpdateFormulas, fit (the main training loop). Another class can implement the activation functions and their derivatives, while a third class can implement the loss function, and so on. Please note that these are just suggestions and you are not required to take this approach. Please feel free to be creative when implementing the MLP.

B. Hyperparameters Optimization (10 points)

1. (2 points) *Parameter initialization.* Choose the best performing model from Part A in terms of classification accuracy. For this model, initialize all its parameters (connection weights and biases) to zero and retrain it using all the other settings from Part A. Report the performance (e.g. loss function over training epochs, classification accuracy and confusion matrix on the validation set after training) of this new trained model. Do you observe any interesting behavior for this new trained model? Discuss the performance of this new trained model in comparison with the performance of the best model from Part A.

2. (2 points) *Learning rate vs parameter initialization.* Retrain several times for the same fixed amount of epochs (e.g. 100 - this value is your choice) the best model from Part A using exactly the same settings as in Part A and the same random seed, with the exception of the learning rate and the initialization of the connection weights and biases which have to be different for each retraining. Start with a very small learning rate (e.g. 0.00001) and after that gradually

increase it (e.g. next values can be 0.0001, 0.001, ...). Initialize the connection weights and biases by sampling from a normal distribution $\mathcal{N}(0, \sigma^2)$[2]. σ^2 has to take the following values $\{0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1\}$. Make a heatmap[3] where the x-axis represents σ^2, the y-axis represents the learning rate, while the colors represent the accuracy[4] obtained on the validation set after each training. Each element in the heatmap matrix represents practically the accuracy obtained with a specific learning rate and a specific initialization. Discuss the heatmap results.

3. (2 points) *Activations.* Choose the worst (one which still trains and does not give errors during training) and the best performing models from Subproblem B.2 in terms of classification accuracy. For both models, visualize the output of each hidden neuron (after applying the activation function) from the two hidden layer three times (after connection initialization before to start the training process, at half of the training process, and at the end of the training process). Visualization hints (it is optional to follow them, please feel free to be creative): you can compute the hidden neuron ouputs on the validation set where the data points are ordered per class; you can use a heatmap where x-axis represents the hidden neuron id, the y-axis represents the data points id, and the colors represent the hidden neuron ouputs. Discuss the results.

4. (4 points) *Hyperparameters Optimization.* Train a MLP model on which you perform hyperparameters optimization. For this specific subproblem, you are not constraint any-more to use just two hidden layers, and 10 neurons per hidden layer. You are free to vary any hyperparameter (e.g. learning rate, number of hidden neurons per layer, type of activation function) you consider to be important to maximize the accuracy on the validation set. Please make a Parallel Coordinates Plot[5] to study the effect of the hyperparameters choice on the performance. The colors of the lines which connect various hyperparameter settings represent the accuracy on the validation set. Discuss the results. For the worst and best performing models, please make a plot with the values of the loss function computed separately over the training and validation sets respectively. Discuss the generalization performance (non exclusive suggestions: overfitting, underfitting) of these two extreme models.

   You can find an example for classification accuracy of a Parallel Coordinates Plot here
   https://docs.microsoft.com/en-us/azure/machine-learning/service/how-to-tune-hyperparameters

C. Peer Review paragraph (0 points)

Finally, each group member must write a single paragraph outlining their opinion on the work distribution within the group for this assignment (THA3). Did every group member contribute equally? Did you split up tasks in a fair manner, or jointly worked through the exercises. Do you think that some members of your group deserve a different grade from others?

Success!

[2] https://numpy.org/doc/stable/reference/random/generated/numpy.random.normal.html
[3] https://en.wikipedia.org/wiki/Heat_map
[4] For convenience, please use 0 for the cases when Python gives error during training.
[5] https://en.wikipedia.org/wiki/Parallel_coordinates