Técnicas e Algoritmos em Ciência de Dados
Assessed Coursework 2

This assignment must be submitted by June 28th, 2021 at 8:00 am.
Late submissions will NOT be accepted.

Learning outcomes assessed

This coursework will test the fundamentals of neural network training, decision trees bagging, and random forests implementation.

Instructions

Identifier

Please choose a random number of 6 digits and write it in the first cell of the notebook. Make sure that you keep a copy of that number as it will be used to provide the feedback (please avoid trivial numbers, such as 000000 or 123456, or identifiers starting with zero – thank you).

Submission

Submit your files through ECLASS. The files you submit cannot be overwritten by anyone else, and they cannot be read by any other student. You can, however, overwrite your submission as often as you like, by resubmitting, though only the last version submitted will be kept. Submission after the deadline will NOT be accepted.

If you have issues, at the very last minute, email your coursework as an attachment at alberto.paccanaro@fgv.br with the subject "URGENT – COURSEWORK 2 SUBMISSION". In the body of the message, explain the reason for not submitting through ECLASS.

IMPORTANT

- Your submission will consist of a single Python notebook implementing your solutions. If you participate in the bonus challenge (explained below), a zip file containing the Python notebook together with the prediction file must be submitted instead.
- The name of the file will be the random number that identifies you (for example 568423.ipynb)
- This coursework consists of 4 parts. Please make sure that the 4 parts are clearly separated and identifiable in your Python Notebook.
- DO NOT SUBMIT ANY DATASET, only the code.
- Any utility function that you will use should be included in the notebook – do not submit utility scripts.
- Python notebooks can become unordered when writing the code. If a cell in your notebook depends on another cell that is BELOW it, marks will be deducted.

ADVICE ON BONUS EXERCISES

The value of each exercise in percentages is provided – the total sum of the marks for the exercises is 100. However, there is an extra **BONUS CHALLENGE** at the end, that is worth an extra 10%. *Note that this exercise is difficult and time-consuming. I advise you to have a final working solution for the entire coursework before attempting to answer this optional exercise.*

All the work you submit should be solely your own work. Coursework submissions will be checked for this.

Marking Criteria

This coursework is assessed and mandatory and is worth 25% of your total final grade for this course. In order to obtain full marks for each question, you must answer it correctly but also completely. Marks will be given for writing well structure code.

COURSEWORK

Part 1 – Backpropagation – value of this section: 30 %

Download the file “Part1.tsv” from the ECLASS platform. In this part, you will build a neural network with a specific architecture and train it with the backpropagation algorithm that you will implement.

Here are the steps that you will need to implement:

a) Load the data into a pandas DataFrame, and get a scikit-learn compatible dataset.
b) Make a 70%/30% split of the dataset for training and testing respectively.
c) Using numpy, implement the neural network architecture from Figure 1. Note that σ denotes the sigmoid function, *tanh* denotes the hyperbolic tangent and the vertical (single) arrows denote the bias term. You will need to implement the backpropagation algorithm to learn the weights of this network by gradient descent, that is:
   - The forward pass
   - The backward pass
   - Weight updates

   Note: your implementation must rely on numpy only, and it must not use any specialized framework for backpropagation. In other words, your code should look something similar to the one we saw in the backpropagation lab.
d) Evaluate the trained model on your test set using the following:
   - Accuracy
   - AUC-PR
   - AUC-ROC
   - printing the confusion matrix

   Note: In this step, you can use scikit-learn tools, no need to implement these.
e) During each epoch of training, collect the value of the error, and make a plot with the epoch number in the X axis, and the error on the Y axis.

[Hint: because of the specific architecture that you must implement, this cannot be implemented in a straightforward way using scikit-learn. I advise you not to waste time trying.]

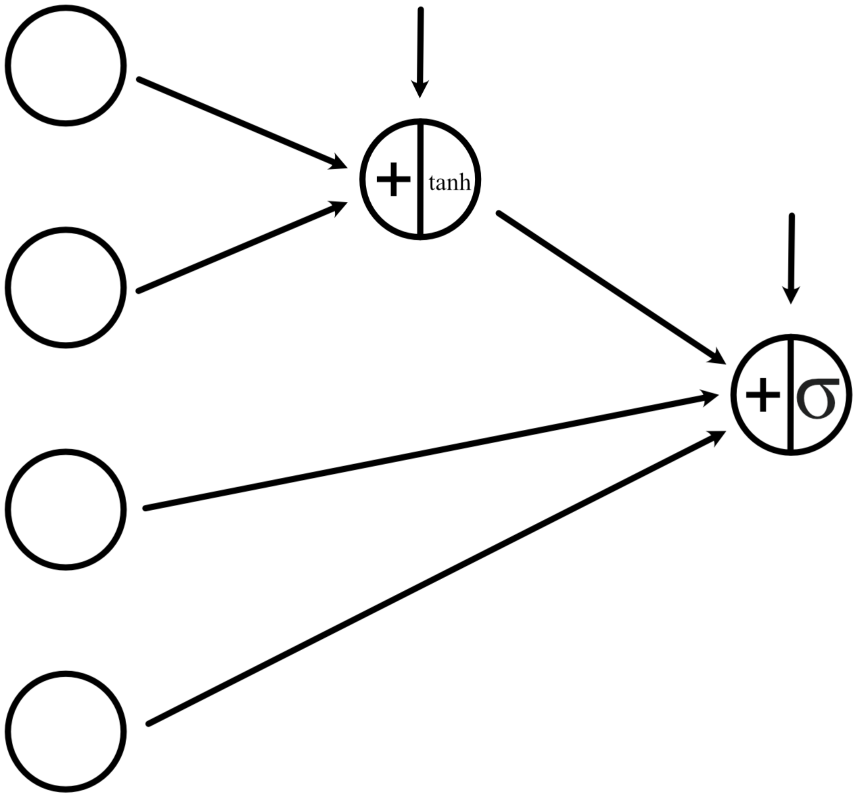

Figure1

Part 2 – Bootstrap aggregation – value of this section: 25 %

Download the Census Income dataset (https://archive.ics.uci.edu/ml/datasets/Adult). In this part, you will build a bagged trees model for classification.

Here are the steps that you will need to implement:

a) Load the data into a pandas DataFrame, and get a scikit-learn compatible dataset.
b) Make a 70%/30% split of the dataset for training and testing respectively.
c) Implement a bagged trees model. The model should have:
   - A parameter B, to decide on the number of bootstrapped training sets.
   - Parameters for controlling the growth of trees. You need to implement at least two of the following:
      i. Maximum level (or depth) of the tree
      ii. Minimum number of observations in a node
      iii. The proportion of classes in the node

[Hint: here be careful that some of the features are numerical (like the example that we saw in the lab) and some are nominal (or categorical).]

Part 3 – Random Forests – value of this section: 30 %

This part uses the same data you used already in Part 2.
Here, you will implement the Random Forest algorithm, as described in Section 8.2.2 of the Witten, James, Hastie & Tibshirani book. You should add the following options:

- A parameter m for the number of predictors (or features) to consider at each split
- A parameter to control the number of trees in the forest
- Parameters to control the growth of trees (could be the same as Part 2c)

Part 4 – Performance comparison – value of this section: 15 %

This part relies on the models you implemented in Parts 2 and 3.

a) Use functions you implemented in Parts 2 and 3 to build:
   - A bagged trees scikit-learn estimator
   - A random forest scikit-learn estimator
b) Using the dataset you built in Part 2, train both models, and compare the results using the following performance metrics:
   - Precision
   - Recall
   - AUC-ROC
   - AUC-PR

Your code will:

a) Print a table with values of the performance metric (in the columns) for each model (in the rows) on the test set.
b) Make a single figure containing 2 subplots. In the first subplot, you will plot the ROC curves for each model. In the second subplot, you will plot the PR curves for each model.

BONUS CHALLENGE – value: 10 % EXTRA MARKS

The “Challenge-train.tsv” file contains data that describe associations between proteins and their function. For the purpose of this exercise, we can think of this problem as a binary classification problem with 6 columns, that you find in the file “Challenge-train.tsv”:

- The first 5 are the features
- The last one is the target of your binary classification (either 0 or 1)

The file “Challenge-test.tsv” has the first 5 columns described above, but the label is omitted.

For this challenge, you will use whatever method you want to train a binary classifier on the file “Challenge-train.tsv” and then use it to generate predictions for each line in the file “Challenge-test.tsv”. You will then submit your predictions. We will test your predictions against the true, correct values.

Note that:

- If you submit binary values, 1 is assumed to be a “positive” prediction, and 0 is assumed to be a “negative” prediction.
- If you submit a numeric prediction, we assume that the score is higher when the model predicts positive labels, and lower when the model predicts negative labels.

Rules of the challenge:

- You submit a plain text file named “prediction-<number>.txt”, where <number> is the random number you chose as your identifier. This file must contain a single number per line, constituting the prediction for the corresponding line in the file “Challenge-test.tsv”
- The first line is the prediction for the first line in “Challenge-test.tsv”
- If you submit fewer lines than those included in the test file, for the remaining lines we will assume a prediction of 0 if your model predicts using binary labels, or the minimum value in your file, if your model predicts using a numeric score.
- If you submit more lines than those included in the test file, the exceeding lines will be ignored.
- You can get disqualified if:
    - You submit more than 1 prediction file.
    - The format of the file is not as described above.
    - If the format of the file prevents us from opening it with a plain text editor.
    - The <number> you used to name the file does not match the one you used for the python notebook submitted.

The procedure for allocating the marks will be as follows:

- The AUC-ROC and AUC-PR metrics will be calculated
- For the AUC-ROC:
    - The top 20% of models will get 5% extra marks
    - The next 20% will get 4% extra marks
    - …
    - The last 20% will get 1% extra marks.
- For the AUC-PR we will use the exact same allocation for the marks as for the AUC-ROC.

Tip 1: The more people join the challenge, the better for everyone (think about this!)
Tip 2: It is impossible to find this dataset online, as these data come from our lab and are still unpublished.