Learning with Artificial Neural Networks

Practical Work 03 – Mice's sleep stages classification with MLP

Class A:
    Professor: Stephan Robert
    Assistants: Benoît Hohl, Shabnam Ataee
Class B:
    Professor: Andres Perez-Uribe
    Assistants: Hector-Fabio Satizabal, Simon Walther
Emails : prenom.nom@heig-vd.ch

Goals:

- Prepare and pre-process the available data to train a neural-network for solving a classification task.
- Design experiments to exploit the available data to come up with an automatic classification system.
- Design, train and evaluate neural networks (Multi-Layer Perceptron) to solve a real-world problem.
- Perform hyper-parameter tuning to find an appropriate neural network architecture and to achieve a good performing model.

1. Introduction

In this practical work, you will use MLPs with Keras to classify mice's sleep stages.
We will use the same kind of data that in previous practical works. We have one more mouse for training and another one for testing:

Training data:
EEG_mouse_data_1.csv (same mouse as PW1)
EEG_mouse_data_2.csv (another mouse for training)
Note: if you want to concatenate pandas dataframes, you can do it with « pd.concat ».

Testing data:
EEG_mouse_data_test.csv (another mouse for testing)

All of these mice are genetical clones of each other so the data will be slightly different from mouse to mouse but will still be similar.

We don't provide any notebook. You can use code of previous practical works to help you. You can choose whether you want to do it on Colab or not.

Preprocessing:
- Choose 25 features
- Normalize data
  see: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html

Validation:
- Use 3-fold cross-validation and shuffle data
  Note: we shuffle the data but it's not a perfect solution. In that way, with two data points that are few seconds apart, one could be in training set and the other one in validation, which means that the validation will be highly correlated with the training and would not represent reliably the model's performance on unknown data. If we split data without shuffling, we would have folds with nearly only awake state which would not be desirable as well.

Training history:
- Plot training and validation loss

Performance metrics:
- Confusion matrix
- F1-score (of each class and 'micro', there is a parameter for « micro », check sklearn)

2. Report

Don't include answers to « Questions to ask yourselves » in your report.

For each experiment and for the competition, include your model summary, your performances results, your training history plot, and analyse your results (Is there overfitting? Should we train for longer? What can we say about the confusions the model makes? How did you choose the learning rate? How did you choose the number of neurons and number of layers? …)

For the competition part, explain briefly one of the idea you chose and if you did some other modifications, just state them in your report.

3. First experiment: awake / asleep

Here we will classify between only two states: « awake » and « asleep ».
You must group n-rem and rem sleep stages into a class « asleep ».

Questions to ask yourselves:
- Is it correct to normalize before splitting before training and validation? Why?

4. Second experiment: awake / n-rem / rem

Classify between the three states « awake », « n-rem » and « rem ».

Questions to ask yourselves:

- What do you need to change compared to the previous experiment?
- How many outputs neurons should we have?
- How to encode the model's output when there's more than two classes? (take a look at one-hot encoding)
- Is the target encoding different if you use « tanh » instead of « sigmoid » for the outputs activation function?
- How to decide which class is predicted?
- If you have « tanh » or « sigmoid » as the output activation function and if you decide which class is predicted with a threshold (e.g. > 0 or > 0.5):
  - Is it possible:
    - That multiple class are predicted
    - That no class is predicted
  - What could we do instead of using a threshold?

5. Competition: awake / n-rem / rem

Same experiment as the second one but this time you're free to do any modifications you want as long as it's still an MLP. You can change how you validate the model if you want. Just make sure to give the important details in your report.

Join your results on the test data (make sure to preprocess the test data the same way you preprocessed your training data before predicting on it).

The first three groups will have a bonus.

What you need to do**:**

- Implement at least one new idea.
- You don't need to apply more than one idea if you don't want to. This competition is there to motivate you, not to burden you with a lot of work. Only the top 3 winners will appear in the ranking, but we will inform you of your score and rank privately.
- It can be ideas from this list but can also be other ideas you find yourselves.

Ideas**:**

- Change optimizer (https://keras.io/api/optimizers/) (e.g. Adam instead of SGD)
- Change last layer activation function by « softmax » (for each output of the model we will predict a *probability* and the sum of the output values equal to 1).
  - Use CategoricalCrossentropy (which is a loss meant for classification, https://keras.io/api/losses/probabilistic_losses/#categoricalcrossentropy-class) instead of « MSE »
- Use an ensemble (https://machinelearningmastery.com/model-averaging-ensemble-for-deep-learning-neural-networks/) of models (train multiple models, average their predictions)
- …

6. Submission

Provide your report, your notebook(s) and your prediction on the competition test set.
Report must be a **PDF**. Don't forget to include all your group members names in the report!

Competition's predictions:
Save your results in a file named « test_pred.npy » and add it to your submission. Make sure the order of your predictions is correct (don't shuffle test set…).

test_predictions must be a numpy array with **the name** of the sleep stages you predicted.
We cannot know what sleep stage you refer to if you have them as integer!

Example: array(['n', 'w', 'r', …])

```
with open('test_pred.npy', 'wb') as f:
        np.save(f, test_predictions)
```