Machine Learning A

2023-2024

Home Assignment 5

Christian Igel
Department of Computer Science
University of Copenhagen

The deadline for this assignment is **5 October 2023, 22:00**. You must submit your *individual* solution electronically via the Absalon home page.

A solution consists of:

- A PDF file with detailed answers to the questions, which may include graphs and tables if needed. Do *not* include your full source code in the PDF file, only selected lines if you are asked to do so.
- A .zip file with all your solution source code with comments about the major steps involved in each question (see below). Source code must be submitted in the original file format, not as PDF. The programming language of the course is Python.
- IMPORTANT: Do NOT zip the PDF file, since zipped files cannot be opened in *SpeedGrader*. Zipped PDF submissions will not be graded.
- Your PDF report should be self-sufficient. I.e., it should be possible to grade it without opening the .zip file. We do not guarantee opening the .zip file when grading.
- Your code should be structured such that there is one main file (or one main file per question) that we can run to reproduce all the results presented in your report. This main file can, if you like, call other files with functions, classes, etc.
- Handwritten solutions will not be accepted. Please use the provided latex template to write your report.

1 Sleep well (35 points)

Sleep is one of the most fundamental physiological processes, and abnormal sleeping patterns are associated with poor health. They may, for example, indicate brain- & heart diseases, obesity and diabetes. During sleep our brain goes through a series of changes between different *sleep stages*, which are characterized by specific brain and body activity patterns. *Sleep staging* refers to the process of mapping these transitions over a night of sleep. This is of fundamental importance in sleep medicine, because the sleep patterns combined with other variables provide the basis for diagnosing many sleep related disorders (Kales and Rechtschaffen, 1968, Iber and AASM, 2007). The stages can be determined by measuring the neuronal activity in the cerebral cortex (via electroencephalography, EEG), eye movements (via electrooculography, EOG), and/or the activity of facial muscles (via electromyography, EMG) in a *polysomnography* (PSG) study. The classification into stages is done manually. This is a difficult and time-consuming process, in which expert clinicians inspect and segment the typically 8–24 hours long multi-channel signals. Contiguous, fixed-length intervals of 30 seconds are considered, and each of these *segments* is classified individually. Algorithmic sleep staging aims at automating this process. The state-of-the-art in algorithmic sleep staging is marked by deep neural networks, which can be highly accurate and robust, even compared to human performance, see the recent work by Perslev et al. (2019) and references therein.

This assignment considers algorithmic sleep staging. The data is based on a single EEG channel from the Sleep-EDF-15 data set (Kemp et al., 2000, Goldberger et al., 2000). The input is given by an intermediate representation from the U-Time neural network architecture (Perslev et al., 2019), the targets are sleep stages. We created a training and test set, the inputs and the corresponding labels can be found in `X_train.csv` and `y_train.csv` and `X_test.csv` and `y_test.csv`, respectively.

1.1 Data understanding and preprocessing

Download and extract the data from `https://github.com/christian-igel/ML/blob/main/data/Sleep-EDF-15_U-Time/`.

Consider the training data `X_train.csv` and the corresponding labels `y_train.csv`. Report the class frequencies, that is, for each of the 5 classes report the number of data points divided by the total number of data points) in the training data.

The ith row in `X_train.csv` are the features of the ith training pattern. The class label of the ith pattern is given in the ith row of `y_train.csv`.

Deliverables: description of software used; frequency of classes

1.2 Classification

The task is to evaluate several multi-class classifiers on the data. Build the models using the training data only. The test data must only be used for final evaluation.

1. Apply multi-nominal logistic regression. If you use regularization, describe the type of regularization you used. Report training and test loss (in terms of 0-1 loss).
2. Apply random forests with 50, 100, and 200 trees. Report training and test loss (in terms of 0-1 loss).
3. Apply k-nearest-neighbor classification. Use cross-validation to determine the number of neighbors. Report training and test loss (in terms of 0-1 loss). Describe how you determined the number of neighbors.

Deliverables: description of software used; training and test errors; description of regularization and model selection process

2 Invariance and normalization (30 points)

Normalizing each component to zero mean and variance one (measured on the training set) is a common preprocessing step, which can remove undesired biases due to different scaling, see section 9.1 in e-Chapter 9 of the textbook by Abu-Mostafa et al. (2015). Using this normalization affects different classification methods differently.

- Is nearest neighbor classification affected by this type of normalization? If your answer is yes, give an example. If your answer is no, provide convincing arguments why not.
- Is random forrest classification affected by this normalization? If your answer is yes, give an example. If your answer is no, provide convincing arguments why not.

Comment: If a transformation of the input (e.g., component-wise normalization or flipping and rotation of input images) does not change the behaviour of a classifier, then we say that the classifier is invariant under this transformation. When devising a machine learning algorithms for a given task, invariance properties can be an important design/selection criterion. If we know that the prediction of an input should not change if we a apply a certain transformation to it, then it is a plus if an algorithm is invariant under this transformation – the generalization from an input to its transformed version(s) is directly given and need not be learnt.

3 Differentiable programming (35 points)

This task should make you more familiar with PyTorch, automatic differentiation, and iterative gradient-based optimization. These are basics of deep learning, but differential programming is useful in scientific modelling beyond neural networks.

3.1 Steepest descent

Consider the notebook `Gradient example.ipynb` made available in the context of the lecture on linear classification. Your task is to migrate it to PyTorch and use autograd instead of specifying the gradient of the function explicitly. Use the Torch optimizers. Reproduce the plot created by `Gradient example.ipynb`.

Deliverables: show the modified parts of the code in the report

Hints:

- In the beginning, you need to define the trainable parameters of the function you want to optimize. By setting `requires_grad=True` you inform autograd that it should keep track of the gradient of a tensor. The initialization of the parameters should not be taken into account by autograd. Thus, you have to tell autograd to look away when doing the initialization:

```
x = torch.ones(1, requires_grad=True)
y = torch.ones(1, requires_grad=True)
with torch.no_grad():
    x *= 0.9*r
    y *= 0.8*r
```

  The functions in `torch.nn.init` are provided to initialize neural network parameters, and they do the above under the hood. For example, you could alternatively use:

```
x = torch.empty(1, requires_grad=True)
y = torch.empty(1, requires_grad=True)
torch.nn.init.constant_(x, 0.9*r)
torch.nn.init.constant_(y, 0.8*r)
```

- For a tensor with a single element, you can get the value in standard Python by calling `torch.Tensor.item()`.
- For steepest descent use `torch.optim.SGD([x, y], lr=eta)`.
- Do not forget to reset the gradients of your optimizer in the optimization loop.
- A plot of the optimization steps should look like in the example not using PyTorch – so you can check whether your solution is correct.

3.2 Adam [optional]

Replace steepest descent with Adam (Kingma and Ba, 2014) using `torch.optim.Adam([x, y], lr=eta)` with learning rate 0.5. Show a plot of the update steps as in the notebook. Why does the fourth step go in the "wrong" direction?

References

Y. S. Abu-Mostafa, M. Magdon-Ismail, and H.-T. Lin. *Learning from Data. Dynamic E-Chapters.* AMLbook, 2015.

A. L. Goldberger, L. A. N. Amaral, L. Glass, J. M. Hausdorff, P. Ch. Ivanov, R. G. Mark, J. E. Mietus, G. B. Moody, C.-K. Peng, and H. E. Stanley. PhysioBank, PhysioToolkit, and PhysioNet: Components of a new research resource for complex physiologic signals. *Circulation*, 101(23):e215–e220, 2000.

C. Iber and AASM. *The AASM manual for the scoring of sleep and associated events: rules, terminology and technical specifications.* American Academy of Sleep Medicine, Westchester, I. L., 2007.

A. Kales and A. Rechtschaffen. *A manual of standardized terminology, techniques and scoring system for sleep stages of human subjects. Allan Rechtschaffen and Anthony Kales, editors.* U. S. National Institute of Neurological Diseases and Blindness, Neurological Information Network Bethesda, Md, 1968.

B. Kemp, A. H. Zwinderman, B. Tuk, H. A. C. Kamphuisen, and J. J. L. Oberye. Analysis of a sleep-dependent neuronal feedback loop: the slow-wave microcontinuity of the EEG. *IEEE Transactions on Biomedical Engineering*, 47(9): 1185–1194, 2000.

D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2014.

M. Perslev, M. Hejselbak Jensen, S. Darkner, P. J. Jennum, and C. Igel. U-time: A fully convolutional network for time series segmentation applied to sleep staging. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.