Machine Learning A

2025-2026

Preparatory Exercises 4

Christian Igel **Sadegh Talebi** **Yevgeny Seldin**

Department of Computer Science
University of Copenhagen

1 Basic Assumptions

(i) When deriving generalization bounds, we made some basic assumptions about data. What are these assumptions?

(ii) Briefly explain what happens when these assumptions break.

2 Random Forests

2.1 Analyzing Satellite Data

Figure 1: Landsat 8: Example image (true color) that is derived from the multi-spectral bands.

In the first part of this exercise, you will deal with data from the *Landsat 8* satellite. The satellite takes images via *multiple bands*, which yields so-called *multi-spectral images* with multiple values given for each pixel. Using such multi-spectral images, one can generate "normal" true color images, see Figure 1.[1] On Absalon, you can find three preprocessed Landsat 8 files: `landsat_train.csv`, `landsat_validation.csv`, and `landsat_area.csv`. The training file contains $n = 5,000,000$ lines. Each line contains a label (first value) and $d = 9$ features separated via commas (each row corresponds to a pixel in an associated multi-spectral image). The validation file is of the same form and contains $1,335,558$ instances. The last file contains $9,000,000$ lines each 9 features (i.e., no label).

1. Train a random forest with 10 trees and bootstrap samples using the training data. At each node, test all the d features and consider the Gini index as evaluation criterion. Build full trees, i.e., do not set any maximum depth for the trees (which is the default for random forests). Afterwards, compute the validation accuracy using the instances provided in `landsat_validation.csv`. What is the validation accuracy?

2. Next, apply the model to all instances given in `landsat_area.csv` and visualize the predictions (e.g., one color per class). Here, each line of `landsat_area.csv` corresponds to a pixel in a 3000×3000 image, where the first 3000 lines correspond to the first row, the following 3000 ones to the second row, and so on. Do you recognize the area?

3. Assume you are given n training points and that you have built a binary decision tree based on these points (full tree, i.e., you only stop once each leaf is pure). What is the maximum depth/height of such a tree, i.e., how many nodes might be, in the worst case, on the path from the root to a leaf?

Deliverables: Provide all your source code and add answers to the questions raised above. Also include the visualization/plot to your write-up.

Hints: Make use of Python and the Numpy package to load the data. To train the model, make use of the `RandomForestClassifier` *class provided by the Scikit-Learn package. Note that loading the data and training/applying the model* ***might take some time*** *(maybe a few minutes). If you do not have access to a computer that is powerful enough for this exercise (e.g., lacks main memory), consider Google Colab:* `https://colab.research.google.com/`.

[1] If you are interested, you can check out the details related to this type of data, but it's not needed for doing the assignment: `https://landsat.gsfc.nasa.gov/data/`

2.2 Normalization

As discussed, 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. 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 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.