Exercise for MA-INF 2213 Computer Vision SS22
22.04.2022
Submission deadline: 13.05.2022

Important: Use Python 3.8 for your solutions. You can use OpenCV only for reading images, and OpenCV or Matplotlib for visualization. You are not allowed to use any additional python modules beyond the ones imported in the templates. Otherwise you won't get any points. Code with runtime errors or returning obviously rubbish results (e.g. nans, inf, meaningless visual output in future exercises) will give you at most half of the points. Points are also assigned to questions inside the coding exercises. You can complete the exercise in groups of two, but only one submission per group is allowed. Include a *readme.txt* file with your group members into each solution. Points for solutions without a *readme.txt* file will only be given to the uploader.

Decision Forests: An Application to Image Segmentation

The problem of image segmentation can be formulated as a classification task where every pixel in an image is mapped to an already known class C based on the information extracted from its $N \times N$ neighborhoods. In this exercise we aim to learn such a mapping using Decision Forests. A set of images ($img\_\%d.bmp$) and their ground truth segmentation maps ($img\_\%dsegMap.bmp$) is provided to train and evaluate the forests. Each pixel in an image is labeled with an unique color class $C \in (Grass = 3, Water = 2, Sheep = 1, Others = 0)$ in its segmentation map. Separate *train_images.txt* and *test_images.txt* files are provided to specify training and test images respectively. The headers are arranged as [numImages numClasses] and each row contains the name of an image followed by its segmentation map.

1. **Training Samples:** Generate a set of training samples for each class by randomly extracting patches P_M of size 16×16 from each training image (make sure to have a balanced dataset). *(1 Points)*

2. **Training of Decision Tree:** Train a decision tree T with a depth $D = 15$ that maximizes the information gain based on entropy over the training samples. For splitting the data at each node, use a simple threshold based split functions $S_\gamma(P)$ defined on a patch P:

$$S_\gamma(P) = \begin{cases} 1, & F_P^f(q) < \tau \\ 0, & otherwise \end{cases}$$

where $\gamma = \{q, f, \tau\}$ describes the pixel location q within the patch boundaries, the selected color channel $f \in \{R, G, B\}$ of the patch and the defined threshold τ respectively. The minimum number of patches at a leaf node should be at least 20. To obtain binary tests, use 10 random values for color channel, 100 pixel locations, and 50 threshold values (total 50,000 binary tests at each split node). Finally, store the probabilities of each class at the leaf node to be used for classification. *(8 Points)*

3. **Image Segmentation:** Use the trained tree to segment the provided test images such that it classifies each pixel using the sliding window approach where the window size is equal to the patch size. Visualize the segmentation results as a segmentation map where each class is assigned with an unique color (Grass = Green, Water = Blue, Sheep = Red, Others = Black). Comment (in a separate text file) on the performance of your tree. What could be the potential reason behind not achieving excellent/good segmentation results? *(4 Points)*

4. **Decision Forest:** Train a decision forest with 5 trees. Perform segmentation using the trained forest and compare the results with that of a single tree. Provide the results for the test image "*img_12.bmp*" using both decision tree and forest. *(2 Points)*

5. **Sherwood - A library for Decision Forests:** Download the source code of the library from [2]. Compile it for demos following the *ReadMe.txt* file and execute the executable "sw". **Note:** to build the cpp version of the library for Linux, you need to comment the line "return result;" within the function "static void Test()" in file *.../Sherwood/cpp/demo/source/Classification.h*, and type *$ make all* under the path *.../Sherwood/cpp* (tested using Ubuntu 16.04 and 18.04). For instruction on how to use it for classification forests use the command "./sw clas".

   Play with both parts of experiment-5 *(exp5_n2.txt, exp5_n4.txt)* that correspond to a classification problem for two and four classes, respectively. Following the instructions vary the following parameters:

   - The depth of the trees.
   - The number of trees in a forest.
   - The number of candidate feature response functions per split node.
   - The number of candidate thresholds per feature response function.

   Comment on how the performance of the forest varies with these parameters. Provide your comments and the resulting images in a separate text file. *(5 Points)*

References

[1] A. Criminisi and J. Shotton. *Decision Forests: for Computer Vision and Medical Image Analysis*. Springer, 2013

[2] `http://research.microsoft.com/en-us/projects/decisionforests/`