

Exercise for MA-INF 2213 Computer Vision SS23

26.04.2023

Submission deadline: 10.05.2023 Exercise: 12.05.2023

Important: Please use **Python 3.8** for your solutions. You are **not** allowed to use any additional python modules beyond the ones imported in the template. Otherwise, you won't get any points.

Grading: Submissions that generate runtime errors or produce obviously rubbish results (e.g. nans, inf, or meaningless visual output in future exercises) will receive at most 50% of the points.

Plagiarism: Plagiarism in any form is prohibited. If your solution contains code copied from any source (e.g. other students, or solutions from the web. This includes ChatGPT!), you will receive **0 points** for the entire exercise sheet.

Submission: 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 in each solution. Points for solutions without a readme file will only be given to the uploader.

Random Forests: An Application to Image Classification

In this exercise, we aim to learn a mapping from a given image I to a label L based on the features extracted from the image using Random Forests. A training set of 400 RGB images and a test set of 40 RGB images (img-%d.jpeg) are provided to train and evaluate the forests. Each image is labeled with a label $L \in (\textit{Shark} = 3, \textit{Seashore} = 2, \textit{Rapeseed} = 1, \textit{Agaric} = 0)$. Separate *train_images.txt* and *test_images.txt* files are provided, where the headers are arranged as [numImages numClasses] and each row contains the name of an image followed by its label.

1. **Extracting image features (integral image):** An Integral image S is an image where each pixel represents the cumulative sum of a corresponding input pixel (x, y) with all pixels above and left of the input pixel:

$$S(x, y) = I(x, y) + S(x - 1, y) + S(x, y - 1) - S(x - 1, y - 1)$$

Implement your own function to calculate the integral image. Calculate the average of all the pixels within a patch P of size 15×15 using the integral image. (2 Points)

2. **Training of Decision Tree:** Train a decision tree with a depth $D = 5$ 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 $h(I, \theta)$ defined on an image I :

$$h(I, \theta) = \begin{cases} 1, & |P_1|^{-1} \sum_{\mathbf{q}_1 \in P_1} I^c(\mathbf{q}_1) - |P_2|^{-1} \sum_{\mathbf{q}_2 \in P_2} I^c(\mathbf{q}_2) < \tau \\ 0, & \text{otherwise} \end{cases}$$

where $\theta = \{\mathbf{q}_1, \mathbf{q}_2, s, c, \tau\}$ describes a random pixel $\mathbf{q}_i = (x_i, y_i)$ of an image, a patch P_i of size $s \times s$ within the image (centered at \mathbf{q}_i), the selected color channel c , and the defined threshold τ respectively. To obtain binary tests, use 10 random values for color channel (i.e. randomly sample 10 times), 100 random pixel pair $(\mathbf{q}_1, \mathbf{q}_2)$, 5 patch sizes and 20 threshold values (total 100,000 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 Classification:** Use the trained tree to classify the provided test images. Comment (in a separate text file) on the performance of your tree. What could be the potential reason behind not achieving excellent/good classification results? (3 Points)
4. **Decision Forest:** Train a decision forest with 5 trees. Perform classification using the trained forest and print the confusion matrix using `sklearn.metrics.confusion_matrix`. Compare and comment on the results with that of a single tree. (3 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` under the path `.../Sherwood/cpp/bin/linux`.

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. (4 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/>