LUT University

LUT Computer Vision and Pattern Recognition Laboratory 2024-11-12

BM40A0702 Pattern Recognition and Machine Learning

Lasse Lensu

Practical Assignment

Digits 3-D

1 Problem Statement

Your task is to develop a *pattern recognition system* for the *recognition of hand-drawn digits* $\{0, 1, ..., 9\}$. The digits have been "drawn" as free-hand strokes in the air with a straight index finger, and the 3-D location information of the finger tip has been collected by using a LeapMotion sensor. The purpose of the system is to recognise the hand-drawn digits, that is, to classify them correctly based on the time series representing the 3-D location of the index finger.

The index finger location data has been collected from a human experiment with the LeapMotion sensor, driver and software development kit, matleap Leap Motion Controller Interface by Jeff Perry, and a Matlab function by the author of this description. The location information of each stroke of the experiment has been stored into separate Matlab and comma-separated values (CSV) files. The file name contains the information on the digit and a number identifying the stroke as follows: `stroke_DIGIT_NUM.{mat|csv}` where `DIGIT` is the digit label and `NUM` is a running identification number for the stroke. For example, `stroke_3_0001.mat` would mean that the instruction to the human subject was to draw digit 3 and the Matlab file is the first sample for this digit.

Relevant notes related to the data are as follows:

- The ground truth, that is, the digit labels assigned to the strokes come from the instruction given to a human subject of the experiment about what to draw.
- No identification information about the human subjects that participated in the experiment is provided.
- The data has been divided into training and test sets. The samples in the training set are randomly selected from the whole data set, thus, the order of the files does not correspond to the order of strokes during the experiment.

Only the training data is published for the system development. It has a balanced distribution over the classes.

Your task is to implement a function `C = digit_classify(testdata)` that classifies a single sample given as an $N \times 3$ matrix `testdata`. The output class `C` should be an integer corresponding to the recognised digit.

It is allowed to use a language of your choice, for example Matlab, Python or Julia. Provided solution should be self-contained and include instructions and/or scripts necessary to setup the environment (i.e. `environment.yml` for Conda and/or `requirements.txt` for Python, some additional installation scripts, use paths relative to the files you provide, etc.). Before submitting the final project, make sure that you provide everything to run it by, for example, testing it on a separate machine or in a new environment. The projects that cannot be run will get no points.

PO BOX 20, FI-53851 LAPPEENRANTA, FINLAND
E-mail: lasse.lensu@lut.fi

2 Requirements

The project includes *programming, documentation* and a *seminar presentation*. The work is meant to be done in groups of at most three students.

To carry out the *programming task*, you must obey the following rules:

- Allowed: The use of "low-level" Matlab toolbox/$\mathcal{X}$ library functions available in a standard installation. In addition, the method implementations prepared by you or your group member for this course can be used.
- Not allowed: The use of "high-level" Matlab/$\mathcal{X}$ library functions such as `classify` or another similar function doing most or all the necessary work, any other source code from some source, or using automated tools such as large language models for generating the code for the implementation.

To prepare the *documentation* of your work, you must obey the following rules:

- Allowed: You can use references for the documentation if you acknowledge them (with a proper citation to the reference) and do not directly copy any text from a reference.
- Not allowed: Use of any material prepared by others (without properly acknowledging the source), direct copying of sentences or their parts from a reference, or using automated tools such as large language models for generating text for the documentation.

By returning the project results in the form of an implementation and document, you assure that i) you acknowledge all sources (no plagiarism), and ii) you have not used any forbidden material nor tools to create/generate them. Any signs of plagiarism will be examined further.

2.1 Function implementation

The classification function must have the name `digit_classify` and it must take parameters and return values in the required way. The following details must be taken into account:

- Feature extraction: Possible function(s) responsible for feature extraction must be called within the `digit_classify` function.
- Training: You may use the training data as you see fit. If the classifier requires training, it is not a good idea to train the method every time the function `digit_classify` is called. Therefore, any parameters set by the training process need to be loaded or hardcoded into the implementation. If your classification technique requires extra parameters, the parameter values must be fixed (by using default values chosen by you) or determined inside the `digit_classify` function.
- Inference: The classification function only needs to take in one sample ($N \times 3$ matrix) to be classified, possibly extract the features and produce a single class label. The computational complexity of your implementation must be practical so that testing its performance with hundreds of samples can be performed in a reasonable time.
- Environment for experiments: The `digit_classify` function and possible other related functions must run in a standard environment. Both the filename and function name have to be correct.
- High-level functions: The use high-level functions of Matlab or $\mathcal{X}$ such as `classify` is not allowed in your solution. You must make your own classifier implementing a classification method.
- Visualisations or debug information: the submitted code should not produce any visualisations or provide profound debug information. (It is good practice to include a debugging flag into the code to enable or disable producing such info.)

Remember to properly comment your codes. Write also a help section to your codes that tells the purpose of the function, usage, and explanation of the parameters. In Matlab, comments following the first line of a function will show when the `help` command is used with the name of the function. You can see an example, if you type following command in Matlab:

```
>> help mean
>> type mean
```

2.2 Documentation

Write a report in English about your project. The documentation should contain a cover page where you give the following information:

1. The course number and name
2. Project title
3. Date
4. The student numbers, names and roles (description of the contribution) of each group member.

Describe the methods used for the feature extraction and classification in such detail that a reader would be able to understand the approach and in principle implement the same kind of functions for the feature extraction and classification based on your documentation and the cited references. Presenting an algorithm and explaining it in words at a high level is a good way to describe the principles of methods. Justify your choices, that is, present the motivations to select the feature extraction technique and classifier, and the procedure to select the possibly needed parameter values for your solution.

Include in the report your classification results with the given (training) data by performing suitable validation. Standard approaches can be used for this purpose.

At the end of your documentation, you should list all the references used. Note that you are allowed to use any references/information you want, but all source code must be written by you or your group member for this project or for this course earlier. The "low-level" functions of the official Matlab toolboxes/$\mathcal{X}$ library can be used.

2.3 Seminar presentation

Digits-3D seminar will be arranged during the last teaching week. In the seminar, the student groups give short presentations about the results and a presentation is required from each group. It must be given by a group member or group members in the seminar room, or if necessary, remotely. Separate instructions will be provided in Moodle to prepare and submit the presentation to Moodle for the seminar.

3 Deadline and submission

The deadline of submitting the results of your work to Moodle is **Monday, 9 December 2024 at 12:15 EET**. The results consist of the document and the classifier implementation. The *requirements* are as follows (*STNUM* is the student number of one of the group members):

1. The *document* is submitted in pdf format with filename *STNUM*`.pdf`.
2. The *implementation* is submitted as a single `zip` package and the file name of the package is *STNUM*`.zip`. The package includes all relevant codes (feature extraction, training, classification and analysis of results). When the package contents are extracted, the result is a single directory *STNUM*. This directory contains `digit_classify` Matlab function/wrapper and all the other files (except the standard Matlab/$\mathcal{X}$ functions) needed to run the classification function.

4 Evaluation

The work will be evaluated based on the submitted report and implementation according to the following criteria:

1. Methods: The documentation contains grounds for selecting the methods and parameter values, and descriptions that enable understanding the solution.
2. Successful implementation:
   (a) The implementation solves the problem and it is possible to run a classification test as instructed.
   (b) The submitted classifiers will be ranked according to their classification performance, that is, the smaller the classification error, the better the classifier. The classification error will be determined using separate test data available only to the organisers. All classes have approximately equal number of samples in the test set. The classifier should perform better than the "baseline", that is, a random classifier.
   (c) The code is properly commented so that the grader can understand the implementation.
3. Results and analysis: The produced results are appropriately presented and analysed in depth to support the conclusions in the report.
4. Documentation: The report includes sections typical for a scientific report and relevant references.

In addition to the criteria described above, the following criteria will be taken into account in the evaluation:

- Novelty of the approach. (If the idea comes from somebody else's work, you must acknowledge the source.)
- Difficulty of implementing the selected approach. (However, the results from the work cannot be empty even in the case of a complex approach.)
- Well-designed demonstration or visualisation.

Each submission will be evaluated according to the course's instructions. *To pass the course, the project result has to reach at least 50% of the maximum.* The project result affects the final grade of the course when the other conditions are met.

5 Notes and hints

The data has been gathered using a LeapMotion sensor. For more information, see LeapMotion.

The location data can contain spurious samples or sample sequences in a stroke. Therefore, data pre-processing can be beneficial.

It is likely that the feature extraction part has a significant effect on the classification performance. Try to either develop a good method yourself, or look for more information in the related literature (and cite the references whenever necessary).

If there are any questions or problems with the assignment description, the data or if you aim to use $\mathcal{X}$ instead of Matlab or Python, contact the person supervising the project before inventing your own interpretations about the instructions or making too radical assumptions.