Processing command-line arguments: module "argparse"

A lot of useful tools for computer users have a form of commands/scripts which are run from command line (`ls` , `dir` , `grep` , …). These tools usually accept command-line arguments which follow the command name (e.g., `ls -la`) and which affect the behavior of the command. Python contains standard module argparse [https://docs.python.org/3/library/argparse.html] (tutorial [https://docs.python.org/3/howto/argparse.html]) which allows us easilly configure possible parameters of a script and process them.

Command-line arguments in digit classification task

The script you have to create should behave according to specifications [/wiki/courses/be5b33kui/semtasks/start#interface_specification]. This can be achieved by configuring the `argparse` module in the following way:

```
import argparse

def setup_arg_parser():
    parser = argparse.ArgumentParser(description='Learn and classify image data.')
    parser.add_argument('train_path', type=str, help='path to the training data directory
    parser.add_argument('test_path', type=str, help='path to the testing data directory')
    mutex_group = parser.add_mutually_exclusive_group(required=True)
    mutex_group.add_argument('-k', type=int,
                             help='run k-NN classifier (if k is 0 the code may decide abo
    mutex_group.add_argument("-b",
                             help="run Naive Bayes classifier", action="store_true")
    parser.add_argument("-o", metavar='filepath',
                        default='classification.dsv',
                        help="path (including the filename) of the output .dsv file with
    return parser
```

When you use this function in a script as follows:

```
    parser = setup_arg_parser()
    args = parser.parse_args()
    print(args.train_path)
```

variable `args` will contain all the information passed to the script via command-line arguments, such that you can work with them easily.

Skeleton of "classifier.py"

The above code can be found in file classifier.py [/wiki/_media/courses/b3b33kui/cviceni/strojove_uceni/classifier.py] which can be used as the skeleton of your solution. If you run the downloaded module as

```
python3 classifier.py -k 3 ./train_data ./test_data
```

you should see the following output:

```
Training data directory: ./train_data
Testing data directory: ./test_data
Output file: classification.dsv
Running k-NN classifier with k=3
```

i.e.,

- paths to directories with training and testing data passed on the command line,
- the name of the output file which was not specified on the command line and `argparse` thus filled in the default value, and
- the chosen classifier type (k-NN) with a numeric argument. (But you have to implement the classifier yourselves. 😃)

courses/be5b33kui/semtasks/05_ml1/argparse.txt · Last modified: 2024/02/18 20:06 by xposik