

Handwritten Digit Recognition

OVERVIEW

In this assignment, you will apply Machine Learning skills. You are asked to design and develop one of its popular classification algorithm called kNN (k-Nearest Neighbors) to programmatically predict handwritten digits in its digitized format. kNN is supervised based algorithm by majority vote of its neighbors. It is supervised in a sense that a sufficient large sample dataset of known target values is carefully selected. This sample dataset will be split into 2 groups. One group will be used to formula a kNN model to be used for prediction. It's called Training. The other group is used to verify the accuracy of the established model, called Testing. The Training-Testing cycle might result in several iterations of tuning and adjustment until a satisfactory model is achieved with high accuracy of positive predictions, higher better. Once such a model has been implemented, it's ready to make prediction. To make a prediction, simply feed an unknown value of same format to the program and then perform the required computation in accordance to the established kNN model.

SCOPE

Note: refer to the sample output for references.

1. Construct a kNN models based on 2 files called digit-training.txt & digit-testing.txt for training & testing respectively (cross-validation is not required), files are text based.
2. Refer to the lab materials for more information on kNN implementation, file format and ideas.
3. Use **ONLY** vector-based function developed from previous lab class to determine nearest neighbors
 - a. **Download the latest version of vector.py from moodle**
4. Implement majority vote algorithms to do the best guess prediction; you need to experiment with different values of "k" (3,5,7 or 9) and apply your majority rule to make the prediction. By comparing the results (accuracy) of each outcome, pick the one with the highest accuracy rate.
5. IN THE DESIGN DOC: explain (a) how the closest neighbors are chosen and (b) the rule(s) used in making the prediction
6. Based on your "FINAL" kNN model implemented (value of k and rule(s)):
 - a. show training and testing info (see Output sections)
 - b. show prediction outcome using file digit-predict.txt (see Output sections).
7. Files to download from moodle:
 - a. digit-training.txt, digit-testing.txt, digit-predict.txt & vector.py

SKILLS

In this assignment, you will be trained on the use of the followings:

- Machine Learning life cycle – dataset, data mining, kNN construction, training & testing
- Python objects & modules (file IO, string, string formatting, sorting, dictionary, list, list comprehensions)
- Controls – if, while, for to control program flow
- Variable Scope
- Functions to breakdown the logic

DELIVERABLES

1. Design documentation (A3_School_StudentID_Design.doc/pdf)
2. Program source code (A3_School_StudentID_Source.py)
3. Output (A3_School_StudentID_Output.doc/pdf)

Zip all files above in a single file (A3_School_StudentID.zip) and submit the zip file by due date to the corresponding assignment folder under “Assignment (submission)”

For instances, a SME student with student ID “119010001”:

- A3_SME_119010001.zip:
 - A3_SME_119010001_Design.doc/pdf
 - A3_SME_119010001_Source.py
 - A3_SME_119010001_Output.doc/pdf

5% will be deducted if any files are incorrectly named!!!

OUTPUT

1. Training Info (see sample output)
2. Testing Info (see sample output)
3. Prediction Outcome (see sample output)

DESIGN DOCUMENTATION

For the design document provide write-up for the following information:

1. Design:
 - a. Describe the general structure of the program (functions, variables and program flow).
 - b. Describe kNN model you implemented:
 - i. your choice of k value
 - ii. how the closest neighbors are determined
 - iii. the rule(s) used in making the prediction
 - c. Propose one strategy in reducing the kNN computation time (finding the neighbors) specific to this assignment; “random” is already suggested in class.
2. Test Plan: (Not Required)

TIPS & HINTS

- Use Dictionary to keep list of digit-vectors (during training) and to track accuracy rate (during testing)
- Use Counter() and most_common() from module “collections” to return the closest neighbors
- Use zip + list comprehension for vector sum, subtract, sum, or, average, sum and so on
- Use String Formatting for training and testing info
- Use reduce() from module “functools” to combine multiple vectors into a single OR-vector or AND-vector, if needed

SAMPLE OUTPUT – TRAINING

```

2 Beginning of Training @ 2017-01-23 19:54:44
3 -----
4 Training Info
5 -----
6 0 = 100
7 1 = 94
8 2 = 93
9 3 = 105
10 4 = 87
11 5 = 81
12 6 = 95
13 7 = 90
14 8 = 109
15 9 = 89
16 -----
17 Total Samples = 943
18 -----

```

SAMPLE OUTPUT – TESTING

```

19 -----
20 Testing Info
21 -----
22 0 = 187, 2, 99%
23 1 = 192, 6, 97%
24 2 = 194, 1, 99%
25 3 = 197, 2, 99%
26 4 = 176, 10, 95%
27 5 = 177, 10, 95%
28 6 = 193, 2, 99%
29 7 = 198, 3, 99%
30 8 = 164, 16, 91%
31 9 = 181, 23, 89%
32 -----
33 Accuracy = 96.12%
34 Correct/Total = 1859/1934
35 -----
36 End of Training @ 2017-01-23 19:59:12

```

SAMPLE ODUTPUT – PREDICTION

Simply output the predicted value, one number per line, such as:

```
9
8
6
6
8
9
6
```

MARKING CRITERIA

- Coding Styles – layout, comments, white spaces, naming convention, variables, indentation.
- Documentation – Design + Test Plan
- Program Correctness – logic, program structure, functions with appropriate parameters
- User Interaction – how informative and accurate information is exchanged between game player and host.
- Readability counts – programs that are well structured and easy-to-follow using functions to breakdown complex problems into smaller cleaner generalized functions are preferred over a function embracing a complex logic with nested conditions and sub-functions! In other words, a design with clean architecture with high readability is the predilection for the course objectives over efficiency.
- KISS approach – Keep It Simple and Straightforward.
- Balance approach – you are not required to come up a very optimized solution. However, take a balance between readability and efficiency with good use of program constructs.

CHALLENGES

Determine other means of reducing kNN computation time yet keeping accuracy rate relatively high.

DUE DATE

May 6th, 2018