CS451/551 – Introduction to Artificial Intelligence
Assignment-3

Hand-written Digit Recognition

created by Furkan Kınlı, for help → e-mail: furkan.kinli@{ozu, ozyegin}.edu.tr
Language: Python

1. Definition

In this assignment, you will implement a framework that uses different Machine Learning algorithms to solve one of the most well-known Computer Vision problem, which is Handwritten Digit Recognition (HDR). For this time, we will NOT provide a baseline code for this problem, and you need to implement the algorithms for solving HDR from scratch. Mainly, we expect you to implement 3 different kinds of classification algorithms (*i.e.* Bayesian-based, Tree-based, Nearest Neighbor-based), and train them with training partition of MNIST dataset that have 60.000 grayscale hand-written digit image, and lastly measure the performance of each model on test partition of MNIST.

In this assignment, it is allowed to use scientific computation libraries such as NumPy, Pandas, Matplotlib and Scikit-learn, and it will decrease your coding efforts a lot. However, we kindly expect you to conduct **a comprehensive experimental setup where the effects of each parameter/methodology for each algorithm can be observed**. The performance metric for this task is classification accuracy.

You will split the training partition of MNIST dataset into training and validation parts, and you are expected to use training part to train (fit) the algorithms, then you will measure the performance of each algorithm on validation part, so you can search the effects of each parameter on the validation performance of the algorithms. At the end, you will test your models with their "tuned" parameter settings on the testing partition of MNIST dataset.

In the report, you are expected to introduce the project and the aim of this project, to explain the algorithms used in this project in detail, to show experimental setup (*i.e.* each parameter that you changed during training) for each algorithm, to report the training performance of all parameter settings and the test performance of tuned parameter settings for each algorithm. Performance metric contains "classification accuracy" and "confusion matrix" Descriptive tables, plots and figures are needed for showing the accuracies for each setup on same plot, for showing the accuracies for each algorithm in same table, for training & test confusion matrices for each algorithm.

Libraries & Docs:

- NumPy: website | documentation
- Pandas: website | documentation
- Matplotlib: website | documentation
- Scikit-learn: website | documentation

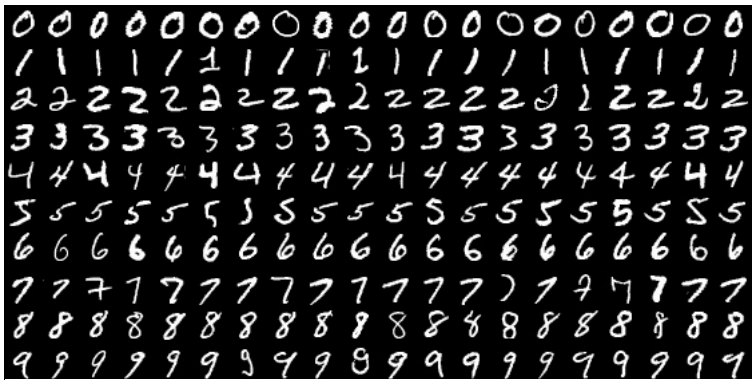
Algorithms to be implemented:

- KNearestNeighbourClassifier: info
- NaïveBayesClassifier: info
- DecisionTreeClassifier: info

Bonus algorithms that may be included to the setup to improve your skills (NOT GRADED):

- KNearestCentroidClassifier: info
- SVMClassifier: info
- SGDClassifier: info
- RandomForestClassifier: info
- MLPClassifier: info

2. MNIST Dataset



The MNIST database (Modified National Institute of Standards and Technology database) is a large database of handwritten digits that is commonly used for training various image processing systems. The database is also widely used for training and testing in the field of machine learning. It was created by "re-mixing" the samples from NIST's original datasets. Furthermore, the black and white images from NIST were normalized to fit into a 28x28 pixel bounding box and anti-aliased, which introduced grayscale levels. This database contains 60,000 training images and 10,000 testing images. The half of the training set and half of the test set were taken from NIST's training dataset, while the other half of the training set and the other half of the test set were taken from NIST's testing dataset. The original creators of the database keep a list of some of the methods tested on it. In their original paper, they use a support-vector machine (SVMs) to get an error rate of 0.8%.

Reference: website | Kaggle | Scikit-learn | Example

I strongly recommend & ask you to read about the algorithms and the problem before starting to implement your assignment. TA will NOT help coding!

4. Implementation Details

For this assignment, we will not provide a baseline code, which means you will implement this framework solving recognizing the digits from scratch. You may use any number of PY files in your code, but **you need to have a single "main.py" that is responsible for running your experiments** in order with training setup and best-performed test setup. Any descriptive comments in the code are welcomed. Please use **only** Scikit-learn version of this database.

As *an advice* for building the framework, for each algorithm, you may have a single PY file containing classes or functions to train and test the algorithm. You may import them into main PY file, and run the experiments with corresponding parameters.

If you are not comfortable to write code in Python, you may follow this tutorial to improve your Python programming skills. Even if you do not know anything about Python, you only need 2-3 days to learn Python from scratch to be able to complete this assignment. If you have any question about the implementation, please do not hesitate to send an e-mail to the assistant.

This algorithm and this problem are well-studied in the literature and online resources, and there are many different coding examples online. We have almost all of them, and please do NOT try to use them directly. In any circumstances of copying online resources directly, you will get 0 point.

5. Criterion

Runtime: Please do NOT include any third-party library to your project (except NumPy, Pandas, Matplotlib and Scikit-learn), because you do not need any extra library. Including some other different third-party libraries may cause a problem for running your code in different local machines (*e.g.* dependencies etc.).

For any dependency (third-party library) on runtime that the assistant can solve → -15 points penalty.

for any dependency on runtime that the assistant cannot solve → -30 points penalty.

Compilation: Please make sure that **your code is compiled properly.**

Not compiling code (*e.g.* any error etc.) → --60 points penalty.

Report: You need to write a report to explain your design in detail. 1 paragraph code explanation is not a report, and such reports will be ignored. Please follow the rules of technical report writing (*e.g.* Introduction, Algorithms, Implementation Details, Results, Conclusion etc.).

Submission: All project files (*.PY) and report file (.PDF) should be zipped (.ZIP) together, and the filename of ZIP file should be in the format of "NAME_SURNAME_ID_hw2.zip". Please follow this structure on your submission. You do **NOT** need to include the data files to your submission.

Not following → -10 points penalty.

Due date: 31 May 2020 – 23.55 via LMS.
Late submission: Allowed for 3 extra days, with -10 for 1st day, -15 for 2nd day, -20 for 3rd day.

Group effort: You may discuss the algorithms and so forth with your friends, but this is an individual work, so you have to submit your original work. **In any circumstances of plagiarism, first, you will fail the course, and the necessary actions will be taken immediately.**

Grading criteria:

- Implementing KNearestNeighborClassifier: 10 points
- Implementing NaïveBayesClassifier: 10 points
- Implementing DecisionTreeClassifier: 10 points
- Report: (70 points)
    - Discussing the effect of "parameters": 40 points (10 for each)
    - Plotting & Tabling training results (accuracy & confusion matrix): 15 points
    - Plotting & Tabling test results with tuned parameters (accuracy & confusion matrix): 15 points