Course name: Machine Learning, Blok4

Course code: 880083-M-6

Academic Year: 2020-2021

Çiçek Güven and Itır Önal
Department of Cognitive Science and Artificial Intelligence
Tilburg University

Machine Learning Assignment

Introduction

For this assignment you will participate in groups of 3 students in a learning challenge. You will need to submit your **predictions**, as well as a report describing your **solutions, analysis** and the **code** which you used to generate the predictions.

The assignment is worth 30% of your grade.

The assignment grade will be based on the quality of your work as judged by the instructor based on your report and code.

Report

Your report should be **4 page maximum** in **PDF format** and should include the following:

- Group number and name of all group members
- Description of your experiments and results including:
  - features used
  - learning model and algorithm used
  - parameter tuning
  - description of method or system built to perform classification
  - discussion of experiments run and performance of your solution (this may include analysis such as confusion matrix, accuracy per class, etc.)
  - details of specification of the work done by group members
  - Reference or appendices (if applicable)

Code

Your code should be a plain python script (.py not a notebook) which can be run to generate your solutions. No data files should be included.

Submission

You can submit your assignment via the canvas assignment submission. The submission must contain the following files:

- Report (.pdf)
- Code (.py)
- Predictions (.csv)

Group work

Your report needs to contain a detailed description of who did what, so make sure to keep track of this information.

Note: it is not acceptable to just say *All members worked together and contributed equally.*

If there are any problems with collaboration, such as serious disagreements, a group member not contributing, or a group dissolving, make sure to inform the course coordinator as soon as possible via email.

Code reuse rules

Remember this is a group assignment. You are **not allowed** to collaborate or share code with students outside your group. Submissions will be checked for plagiarism. If you are found breaking the above rules you will be reported to the Board of Examiners for fraud.

You are **allowed to use**:

- code examples provided by the instructor during the course
- open source libraries available for Python
- open source code found on Github, as long as it is credited in your script and report with a link to the source.

Tasks and Dataset

Image classification (Sign language letters)

In this assignment there are two separate tasks both involving classification of sign language letters.

Maybe create a range for similar gestures.

Figure 1: Images of example sign language letters and their labels

Task 1 - 60 points

The first task is to train two models to recognize sign language letters and compare their performances. You can choose any two models (Neural Networks, multi-class logistic regression, multi-layer perceptron (MLP), Support Vector Machines (SVM), K-Nearest Neighbor, etc.) for comparison. You will split your training data (*train_data_label.npz*) into training and validation sets to train your models and finding the optimal parameters, respectively. You will test the performance of the two models on the same test dataset (*test_data_label.npz*)

You will use:

- **training part** to train your model,
- **validation part** to find the optimal parameters of your model,
- **test part** to evaluate the model with the optimal parameters found on the validation set.

The training and test datasets consist of images of sign language letters and their labels as shown in Figure 2. The labels correspond to the place of the letter in the English alphabet (0-25). Note that there are no cases for J=9 or Z=25 because of gesture motions. For example: A is 0, B is 1, I is 8, K is 10, and so on. In total, there are 24 classes excluding J=9 and Z=25.

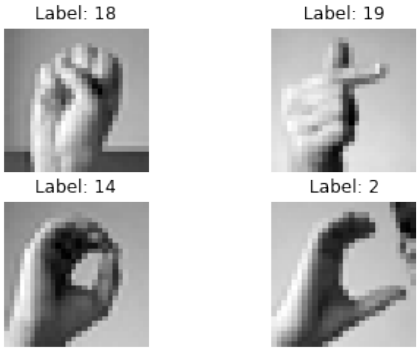

Figure 2: Example images and labels

The training dataset named *train_data_label.npz* and the test dataset named *test_data_label.npz* are available for download on the canvas homepage. This file contains two python lists of (1) images (as numpy arrays) and (2) corresponding labels (indexed by their position in the list). You can load this data by using the code snippet shown in Figure 3.

```
with np.load('train_data_label.npz') as data:
    train_data = data['train_data']
    train_label = data['train_label']

with np.load('test_data_label.npz') as data:
    test_data = data['test_data']
    test_label = data['test_label']
```

Figure 3: Code snippet you can use to load the training and the test data (images and labels). train_data and train_label correspond to training images and labels, respectively. Similarly, test_data and test_label correspond to test images and labels, respectively.

Please make sure you have all the data available to you. Training data consists of **27455 images and labels** in total. Test data consists of **7172 images and labels** in total. Each training and test instance is a 784 dimensional vector, which can be represented as an image of size 28×28 by reshaping.

Task 2 - 40 points

In the second task you will use the classifiers trained in the first task (or you can train new classifiers) to identify a series of n sign language letters in an image of size 28×200. n can take values in the range $[1, 5]$ randomly. Letters are overlaid on a noisy image as shown in the Figure 5.

Figure 4: An example test image. The label for this particular image is '180019'. The letters in the image are: 'SAT'. You can encode/decode the label as: S = 18, A = 00, T = 19, considering the position of the letter in the English alphabet.

Figure 5: Another example test image. The label for this particular image is '0418182407'. The letters in the image are: 'ESSYH'. You can encode/decode the label as: E = 04, S = 18, S = 18, Y = 24, H = 07, considering the position of the letter in the English alphabet.

Note that the label of an image is generated by concatenating the positions of its letters in the English alphabet. For example, if the letters in the image are: 'ABCD', the label for this image will be '00010203'. A dataset of sample test images with their true labels will be provided.

Top-5-accuracy: In this task, top-5-accuracy measure will be used for evaluation. Accordingly, you will make 5 predictions for each image. If 1 out of the 5 predictions is correct (same as the true label) then it is qualified as a correct classification. For example, if the letters in the image are 'abcd' (true label = 00010203) and the predicted labels are:

[01020310 , 0001020304 , 010203 , 02 , 00010203]

then this is classified as a correct prediction since one of the predictions of the classifier is correct.

Test Dataset

test_images_task2.npy is a list of 10000 images. For each image your designed classifier will make 5 predictions. Each row in the *prediction.csv* file will contain 5 predicted labels separated by comma.

Thus the final *prediction.csv* is a 10000×5 file (no header, no index) where k^{th} row represents the predictions corresponding to the k^{th} image in the test set. A snapshot of how the final *prediction.csv* should look like is shown in Figure 6. Make sure you predict the labels of the images in the same order as they are listed in the *test_images_task2.npy*.

As a sanity-check, make sure that the first two images in the *test_images_task2.npy* correspond to labels: '19181012' (TSKM); '2423142413' (YXOYN) and the last image corresponds to label: '231807' (XSH) respectively.

```
21191421,210714,17191421,211913,141914
2012,1812,111218,200818,20
0210,120210,200210,220210,080210
06,0602,0603,13,1002
0713210705,1913210704,07142107,19142107,07032107
181618,10181618,14181618,11161820,0818
07040219,07040819,07150219,07150819,07040207
2308160613,2301160612,2308180613,2308160513,2318160612
```

Figure 6: Example prediction.csv. This example shows **8 rows and 5 columns**. The *prediction.csv* you submit must contain **10000 rows and 5 columns**

Method

There are four important restrictions on the method used:

- The method should be fully automatic, that is, re-running your code should re-create your prediction file.
- The method should not be hard-coded in any way. That is, if a separate set of test-images are provided as input to the code, it should run and predict the labels without any changes in the code.
- Every software component used should be open-source and possible to install locally. This means that you cannot use proprietary closed-source software or access to a web service to carry out any data processing.
- The method should not use any external dataset which overlaps with the provided data. If you wish to make use of external data in your solution ask the instructor via the course forum to confirm that this data is allowed.

Hint

- For task 1, you can use the *train_test_split* function from *sklearn.model_selection* to split your data into training and validation sets.
- For task 2, you can consider using score/probability value outputs of the classifier. You can try using these scores to find the letters and make guesses in the test image.
- Make sure your predictions do not contain 09 or 25 as J=9 and Z=25 are not included in the dataset.

References

[1] Original Dataset: Sign Language MNIST https://www.kaggle.com/datamunge/sign-language-mnist