Logistic Regression and Neural Network benchmark MNIST classification (assignment 4)

Emil Trenckner Jessen

20th May 2021

Contents

1 Introduction

1.1 GitHub repository

Access to the GitHub repository can be found via the link here:

https://github.com/emiltj/cds-visual-exam

The repository contains the individual assignments (including this assignment) a long with READMEs. The READMEs specify cloning the repository, setup (virtual environment installation and data collection). You may also read the information here.

1.2 Getting started

For running my script I'd recommend following the below steps in your bash-terminal (notice the bash scripts are different, depending on your OS). This functions as a setup of the virtual environment, as well as an execution of a bash script that downloads all the data to the data folders respective to the assignments.

Cloning repository and creating virtual environment

Listing 1: bash terminal - MAC/LINUX/WORKER02

```
git clone https://github.com/emiltj/cds-visual-exam.git
cd cds-visual-exam
bash ./create_vis_venv.sh
```

Listing 2: bash terminal - WINDOWS

```
git clone https://github.com/emiltj/cds-visual-exam.git
cd cds-visual-exam
bash ./create_vis_venv_win.sh
```

Retrieving the data

The data is not contained within this repository, considering the sheer size of the data. Using the provided bash script data_download.sh that I have created, the data will be downloaded from a Google Drive folder and automatically placed within the respective assignment directories.

Listing 3: bash terminal

```
bash data_download.sh
```

After cloning the repo, creating the virutal environment and retrieving the data you should be ready to go. Move to the assignment folder and read the README for further instructions (or read a long here).

2 Logistic Regression and Neural Network benchmark MNIST classification (assignment 4)

2.1 Assignment description

Create two command-line tools which can be used to perform a simple classification task on the MNIST data and print the output to the terminal. These scripts can then be used to provide easy-to-understand benchmark scores for evaluating these models.

- Include two scripts; one with neural networks and one with logistic regression
- BONUS
  - Both scripts output a classification report to the terminal and saves it as well
  - Allow the user to determine number and size of layers for the Neural Network
  - Allow the user to determine parameters for the Logistic Regression
  - Allow the user to add an unseen data of any dimensions, process it, and let the classifier classify the new image
  - Allow the user to save the trained models for future use

2.2 Methods

Specifically for this assignment

Prior to training, both scripts had the data min-max scaled to allow for faster processing and better convergence. Both the training and test data was scaled using the values of the training data, to avoid information to flow from the training set to the test set. These scripts allows for simple model benchmarking on the MNIST dataset, which could prove useful if one wanted to test other models for classification of the same data set. Moreover, all the "[BONUS]" features were included. This means that the user is allowed to specify parameters such as number of epochs (-ep) and hidden layers (-H) for the Neural Network (NN), while the arguments for the Logistic Regression (LR) script lets the user specify C-values (-c) and penalty method (-p). The classification reports can be saved if specified using the argument –save, and the user can furthermore predict any new images (regardless of dimensions and colors) using the argument –individual.

On a more general level (this applies to all assignments)

I have tried to as accessible and user-friendly as possible. This has been attempted by the use of:

- Smaller functions. These are intended to solve the sub-tasks of the assignment. This is meant to improve readability of the script, as well as simplifying the use of the script.
- Information prints. Information is printed to the terminal to allow the user to know what is being processed in the background.
- Argparsing. Arguments that let the user determine the behaviour and paths of the script.

2.3 Results and discussion

Classification reports

When looking at the results, the classification benchmarks for the LR classifier and the NN classifier seem to have similar performance, with a macro average f1-scores of .91 for both models.

	0	1	2	3	4	5	6	7	8	9	accuracy	macro avg	weighted avg
precision	0.96	0.93	0.91	0.89	0.91	0.86	0.96	0.93	0.89	0.91	0.91	0.91	0.91
recall	0.97	0.97	0.91	0.88	0.95	0.85	0.95	0.94	0.83	0.89	0.91	0.91	0.919
f1-score	0.97	0.95	0.91	0.88	0.93	0.86	0.95	0.93	0.86	0.90	0.91	0.91	0.91
support	198.0	232.0	194.0	225.0	165.0	171.0	180.0	203.0	198.0	234.0	0.91	2000.0	2000.0

Table 1: Logistic regression classification report

	0	1	2	3	4	5	6	7	8	9	accuracy	macro avg	weighted avg
precision	0.94	0.96	0.91	0.91	0.88	0.90	0.94	0.91	0.89	0.90	0.91	0.91	0.91
recall	0.95	0.97	0.91	0.91	0.92	0.84	0.93	0.93	0.90	0.88	0.91	0.91	0.92
f1-score	0.95	0.96	0.91	0.91	0.90	0.87	0.94	0.92	0.89	0.89	0.91	0.91	0.91
support	198.0	232.0	194.0	225.0	165.0	171.0	180.0	203.0	198.0	234.0	0.91	2000.0	2000.0

Table 2: Neural network classification report

NN's tend to outperform LR classifiers when both data and training time is plentiful and given the right hidden layer structure and parameter settings. As NN classifiers often take longer to train, the default parameters (hidden layer structure and number of epochs) were set to result in low runtimes for faster processing. More training would likely have resulted in the NN outperforming the LR.

When looking at the performances for the individual numbers, it becomes evident that the classifer did not classify equally well for all numbers. 3, 5 and 8 seem to be harder to predict - this is likely due to the similarity that they may have with other numbers (i.e. pixels of 3 and 5 might have a lot of overlap).

When looking at the classification of an individual test image of the number 4 the LR script was capable of classifying it as 4, wheras the NN incorrectly classified it as 5. Again, this is with the default parameters - longer training for the NN may very well have resulted in a better prediction here, also.

In conclusion, the classification reports serve as benchmarks that might be used when evaluating new models for classification purposes: *how well does another model compare in comparison to the relatively simple models used here?*

2.4 Usage

Make sure to follow the instructions in the README.md located at the parent level of the repository, for the required installation of the virtual environment as well as the data download.

Subsequently, use the following code (when within the cds-visual-exam folder):

Listing 4: bash terminal

```
cd assignment_4
source ../cv101/bin/activate # If not already activated
python lr-mnist.py
python nn-mnist.py
```

2.4.1 Optional arguments

lr_mnist.py arguments for commandline to consider

- "-o" "–outname", type = str, default = "classif_report_neural_networks.csv", required = False, help = "str - containing name of classification report")
- "-s" "–save", type = bool, default = True, required = False, help = "bool - specifying whether to save classification report")
- "-i" "–individual", type = str, default = os.path.join("data", "test.png"), required = False, help = "str - specifying a .png file which is to be classified using this logistic regression model.")
- "-p" "–penalty", type = str, default = "none", required = False, help = "str - specifying penalty for the classifier - possible values: n̈one, l̈1, l̈2, ëlasticneẗ")
- "-c" "–c", type = float, default = 1.0, required = False, help = "int - specifying c-parameter for the classifier. recommended values: 0.01, 0.1, 1.0, 10, 100, where lower values mean stronger regularization")

nn_mnist.py arguments for commandline to consider

- "-o" "–outname", type = str, default = "classif _report _neural _networks.csv", Default when not specifying name of outputfile required = False, help = "str - containing name of classification report")

- "-s" "–save", type = bool, default = True, Default when not specifying required = False, help = "bool - specifying whether to save classification report")
- "-i" "–individual", type = str, default = os.path.join("data", "test.png"), required = False, help = "str - specifying a .png file which is to be classified using this neural networks model.")
- "-H" "–hiddenlayers", type = list, default = [8, 16], required = False, help = "list - specifying the hidden layers, each element in the list corresponds to number of nodes in layer. index in list corresponds to hiddenlayer number. E.g. [8, 16]")
- "-e" "–epochs", type = int, default = 50, required = False, help = "int - specifying number of epochs for training the model. Default = 50")