Assignment 2: Classification benchmarks with Logistic Regression and Neural Networks

Cultural Data Science, 2023

Author: Aleksander Moeslund Wael
Student no. 202005192

Assignment notes (Ross)

For this assignment, we'll be writing scripts which classify the Cifar10 dataset.

You should write code which does the following:

1. Load the Cifar10 dataset
2. Preprocess the data (e.g. greyscale, reshape)
3. Train a classifier on the data
4. Save a classification report

You should write one script which does this for a logistic regression classifier and one which does it for a neural network classifier. In both cases, you should use the machine learning tools available via scikit-learn.

Introduction

This repository contains scripts for performing image classifcation on the Cifar10 dataset using logistic regression and a neural network classifier.

Data

The Cifar10 dataset consists of 60K 32x32 colour images in 10 classes, with 6K images per class.

Models

The `src` folder contains two Python scripts, `lr_classifier.py` and `nn_classifier.py`, which provide the pipelines for importing, preprocessing and performing a classification task on the data.

The `lr_classifier.py` script uses multinomial logistic regression to classify the images, whereas the `nn_classifier.py` script uses a multi-layer Perceptron classifier, both models implemented with scikit-learn for Python.

Pipeline

The Python scripts are structured as follows:

1. Import data
2. Preprocess data

3. Load the model
4. Fit the model to the training data
5. Predict test data
6. Print and save a classification report to `out` folder

How to run

NOTE: *Depending on your OS, run either `WIN_*` (on Windows) or `MACL_*` (on MacOS or Linux).*

1. Clone repository to desired directory

```
git clone https://github.com/alekswael/assignment2-image-classification-
cd assignment2-image-classification
```

2. Run setup script

The setup script does the following:

1. Creates a virtual environment for the project
2. Activates the virtual environment
3. Installs the correct versions of the packages required
4. Deactivates the virtual environment

```
bash WIN_setup.sh
```

3. Run pipeline

Run script in a bash terminal.

The script does the following:

1. Activates the virtual environment
2. Runs either `lr_classifier.py` or `nn_classifier.py` located in the `src` folder
3. Deactivates the virtual environment

```
bash WIN_run_lr_classifier.sh
bash WIN_run_nn_classifier.sh
```

Note on model tweaks

Some model parameters can be set through the `argparse` module. However, this requires running the Python script seperately OR altering the `run*.sh` file to include the arguments. The Python scripts are located in the `src` folder. Make sure to activate the environment before running the Python script.

```
lr_classifier.py [-h] [-t TOL] [-s SOLVER] [-m MAX_ITER] [-p PENALTY]

options:
  -h, --help            show this help message and exit
  -t TOL, --tol TOL     Tolerance for stopping criteria. (default: 0.1)
  -s SOLVER, --solver SOLVER
                        Algorithm to use in the optimization problem. Default is
lbfgs. See scikit-learn documentatio for more info. (default: saga)
  -m MAX_ITER, --max_iter MAX_ITER
                        Maximum number of iterations taken for the solvers to
converge. (default: 100)
  -p PENALTY, --penalty PENALTY
                        Specify the norm of the penalty. (default: none)
```

```
nn_classifier.py [-h] [-hls HIDDEN_LAYER_SIZES] [-i MAX_ITER] [-l LEARNING_RATE]
[-s EARLY_STOPPING]

options:
  -h, --help            show this help message and exit
  -hls HIDDEN_LAYER_SIZES, --hidden_layer_sizes HIDDEN_LAYER_SIZES
                        The ith element represents the number of neurons in the
ith hidden layer. If a single layer, DO NOT put a comma. Specify values WITHOUT
SPACES. (default: 64,10)
  -i MAX_ITER, --max_iter MAX_ITER
                        Maximum number of iterations. (default: 70)
  -l LEARNING_RATE, --learning_rate LEARNING_RATE
                        Learning rate schedule for weight updates (default:
adaptive)
  -s EARLY_STOPPING, --early_stopping EARLY_STOPPING
                        Whether to use early stopping to terminate training when
validation score is not improving. (default: True)
```

Repository structure

This repository has the following structure:

```
│   MACL_run_lr_classifier.sh
│   MACL_run_nn_classifier.sh
│   MACL_setup.sh
│   README.md
│   requirements.txt
│   WIN_run_lr_classifier.sh
│   WIN_run_nn_classifier.sh
│   WIN_setup.sh
│
├───out
│
└───src
```

```
        lr_classifier.py
        nn_classifier.py
```

Remarks on findings

When comparing the classification reports, it seems the NN-classifier performs a bit better at 38% acc compared to the LR-classifier at 30% acc, although both performances are somewhat underwhelming compared to chance level (10% acc).

```
LR-classifier
              precision    recall  f1-score   support

    airplane       0.34      0.38      0.36      1000
  automobile       0.36      0.38      0.37      1000
        bird       0.25      0.20      0.22      1000
         cat       0.21      0.15      0.18      1000
        deer       0.24      0.20      0.22      1000
         dog       0.29      0.29      0.29      1000
        frog       0.27      0.30      0.29      1000
       horse       0.29      0.30      0.30      1000
        ship       0.35      0.40      0.37      1000
       truck       0.39      0.45      0.41      1000

    accuracy                           0.31     10000
   macro avg       0.30      0.31      0.30     10000
weighted avg       0.30      0.31      0.30     10000
```

```
NN-classifier
              precision    recall  f1-score   support

    airplane       0.42      0.33      0.37      1000
  automobile       0.43      0.47      0.45      1000
        bird       0.28      0.40      0.33      1000
         cat       0.27      0.16      0.20      1000
        deer       0.32      0.20      0.25      1000
         dog       0.37      0.34      0.35      1000
        frog       0.31      0.53      0.39      1000
       horse       0.45      0.35      0.40      1000
        ship       0.47      0.48      0.47      1000
       truck       0.44      0.46      0.45      1000

    accuracy                           0.37     10000
   macro avg       0.38      0.37      0.37     10000
weighted avg       0.38      0.37      0.37     10000
```