COM6018 Assignment 2

Copyright © 2023 Jon Barker, University of Sheffield. All rights reserved.

3rd December 2023. v1.0

Which Way Up?

Due: 15:00, Friday 15th December 2023

Table of Contents

- 1. Introduction
- 2. The Data
- 3. The Task
- 4. Additional Rules
- 5. Hints on How to Proceed
- 6. Assessment
- 7. Submission

1. Introduction

The task is to build a classifier that can determine the orientation of an image of part of a face. It must take a square region from a face image and report whether it is the right way up, rotated 90-degree right, rotated 90-degree left or upside down. The square regions are of size 90, 50 or 30 pixels and are taken from original images of complete face that are 94 pixels wide by 125 pixels high.

Further details of the data, the task, the assessment criteria and the submission process are given below.

2. The Data

You will be provided with training and evaluation datasets that have been constructed from the 'Labeled Faces in the Wild' dataset. This is the same data that was used in the lab for the person identification task.

2.1. The training data

The training data consists of 10,803 images representing 80% of the people in the full dataset. They are stored in a file `train.full.joblib` (size 484 MB) that you will be able to download. You are also provided with a file `train.small.joblib` that contains a subset of 1/4 of the images that you might find useful to use during development (size 112 MB).

The data can be loaded with,

```
import joblib

images, person_ids = joblib.load('train.full.joblib')
```

The `images` variable will be a NumPy array with shape (10803, 125, 94), i.e., 10803 images of size 125 by 94 pixels. The `person_ids` will be a NumPy array of shape (10803,) containing each image's person ID from the original database. This is not relevant to the task but might be useful for debugging.

2.2. The evaluation data

The evaluation sets consist of 2,000 examples of square sub-images with sides of length 90, 50 or 30 pixels. These have been sampled from 10% of the people in the dataset that do not occur in the training data set. These sub-images have then been rotated by 0, 90, 180 and 270 degrees so that there are an equal number of each orientation (i.e., 500 examples of each orientation).

The sub-images have been reshaped into feature vectors of length 8100, 2500 or 900 pixels respectively for the 90-, 50- and 30-pixel sub-images and stored in the rows of a data matrix, `x_test`, i.e., the matrices are of size 2000 x 8100, 2000 x 2500 or 2000 x 900 respectively for the 2000 sub-images of size 90, 50 and 30 pixels. The corresponding labels are stored in the

vector `y_test` of size 2000 x 1. The labels are integers in the range 0 to 3, where 0 represents the right way up, 1 represents a 90-degree rotation to the left, 2 represents a 180-degree rotation and 3 represents a 90-degree rotation to the right.

The evaluation data is stored in the file `eval1.joblib` which you will be provided with. You can load this file with,

```
import joblib

data = joblib.load('eval1.joblib')
```

The `data` variable will be a dictionary of dictionaries with the following structure:

```
data = {
    '90': {
        'x_test': <2000 x 8100 matrix of 90 pixel sub-images>,
        'y_test': <2000 x 1 vector of labels>
    },
    '50': {
        'x_text': <2000 x 2500 matrix of 50 pixel sub-images>,
        'y_test': <2000 x 1 vector of labels>
    },
    '30': {
        'x_test': <2000 x 900 matrix of 30 pixel sub-images>,
        'y_test': <2000 x 1 vector of labels>
    }
}
```

For example, to retrieve the feature vectors and labels for the 90-pixel sub-images you would use,

```
x_test = data['90']['x_test']
y_test = data['90']['y_test']
```

2.3. The baseline models

You are provided with example 'baseline' models that use PCA and 1-nearest neighbour

classifier. These models are called knn.30.joblib, knn.50.joblib and knn.90.joblib.

The performance of these models on the evaluation data is as follows:

Sub-image size	% Correct
30	47.6
50	71.7
90	93.2

This is the performance that you should aim to beat.

3. The Task

3.1 Developing a Classification Model

You are asked to make three separate scikit-learn models which can be used to classify the 30, 50 and 90-pixel sub-images respectively. These models should be saved with joblib to files called `model.30.joblib`, `model.50.joblib` and `model.90.joblib` which you will submit. You are provided with a script `evaluate.py` which you can use to check that your models work, and which will be used to assess your system.

The evaluation script can be run with,

```
python evaluate.py <MODEL_FILE_NAME> <IMAGE_SIZE>
```

So you can test your model files with,

```
python evaluate.py model.90.joblib 90
python evaluate.py model.50.joblib 50
python evaluate.py model.30.joblib 30
```

Your model files should be created by a Python script called `train.py` that you will need to write and submit. It will take the training data as input and save the trained model to a joblib file. The `train.py` script should be run with

```
python train.py <TRAINING_DATA_FILE_NAME> <MODEL_FILE_NAME> <IMAGE_SIZE>
```

So, for example, you can train your model files with

```
python train.py train.full.joblib model.90.joblib 90
python train.py train.full.joblib model.50.joblib 50
python train.py train.full.joblib model.30.joblib 30
```

3.2. Writing a Report

You are asked to write a report of you work which you will submit as a PDF file called `report.pdf`. The report should be no more than 2 pages long and should contain the following sections:

- **Introduction** - a brief description the orientation detection problem that you are trying to solve.
- **System Description** - a complete description of your classification system pipeline. In particular, highlight the critical hyper-parameters that you will need to optimise.
- **Experiments** - a description of the experiments you have carried out to tune your system's hyper-parameters. Also be clear about how you have constructed your training dataset. Include results that you have used to justify your final choice of design and hyper-parameters settings.
- **Results and Analysis** - a table reporting the performance of your final system when running on the evaluation data provided. Compare this to the performance of the baseline system that was provided.
- **Conclusions** - a brief summary of your work and a discussion of any conclusions you can draw from your results. What could you do to improve your system further if you had more time?
- **References** - a list of any references you have cited in your report.

The report should be written in a clear and concise style. It should be well-structured and easy to read. It should be written in the third person and should not include any personal pronouns (e.g., I, we, you, etc.). You must generate it using LaTeX. A LaTex template will be provided.

4. Additional Rules

Some additional rules have been set to ensure that the assignment is fair for everyone. Please read carefully,

- Your model files should be named 'model.90.joblib', 'model.50.joblib' and 'model.30.joblib' and must not exceed 20 MB in size. (To be confirmed - check the final version of assignment description.)
- You can only train your models using the training data provided.
- You cannot use any pre-trained models.
- You can only use the standard Python libraries and the following libraries: numpy, matplotlib, seaborn, pandas, scikit-learn, joblib.

This list may be updated/clarified before the final version of the assignment description is released.

5. Hints on How to Proceed

First step

The very first step will be to prepare training data in an appropriate format for training your classifier.

You have only been provided with complete, upright images of faces in the training set. So, you will need to generate a set of randomly rotated sub-images from these complete images, and a corresponding set of labels (i.e., the integers 0, 1, 2 and 3 which represent the orientation).

You will find the NumPy function `np.rot90` useful for generating the rotated sub-images.

Separate data preparation and training scripts

I recommend writing the code as two separate scripts.

- `prepare_data.py` - Takes the training data that has been provided, generates all the rotated sub-images and labels and saves them to a file.
- `train_model.py` - Loads the data generated by `prepare_data.py`, trains a

classifier and saves the trained model to a file.

Splitting it into two scripts will mean that you do not have to generate the data every time you want to train a new model. However, there are other approaches that you could take and it is entirely up to you how you organise your code.

Start simple

I recommend starting out by making a small training set and using a simple classifier, such as the k-nearest neighbour classifier. Make sure that you can get this to work, i.e., to replicate the results that are achieved with the baseline model that has been provided. You can then start experimenting with larger training data sets and more complex classifiers.

6. Assessment

The assignment is worth 60% of the module mark and will be marked out of 60.

The final mark will be based on the following criteria:

- The quality and clarity of your code (20/60)
- The quality and clarity of the written report (30/60)
- The performance of your classifier on the hidden evaluation data (10/60)

Further details of the assessment criteria are given below.

6.1. Code (20/60)

Marks will be awarded based on the following rubric:

Score	Description
Fail (0-9)	The code lacks clarity, organization, and proper documentation. It may have significant errors, making it challenging to understand or run. Essential machine learning concepts are not demonstrated.
Pass (10-11)	The code meets the minimum requirements and demonstrates a basic understanding of scikit-learn. However, the organization and documentation could be improved for better clarity.

Score	Description
Merit (12-13)	The code meets all requirements and shows a good understanding of scikit-learn. It is well-organized, with clear documentation that aids in understanding the implementation.
Distinction (14-15)	The code exceeds expectations, demonstrating a thorough understanding of scikit-learn. It is well-organized, follows best practices, and includes comprehensive documentation that enhances clarity.
Distinction (16-17)	The code is of exceptional quality, showcasing a high level of mastery. It not only meets all requirements but does so with exceptional clarity, well-structured organization, and detailed documentation.
Exceptional (18-20)	The code is flawless. It demonstrates outstanding clarity, organization, and documentation. The implementation goes beyond the requirements, showcasing creativity and innovation in classifier design.

6.2. Report (30/60)

Marks will be awarded based on the following rubric:

Score	Description
Fail (0-15)	The report lacks a clear and coherent description of the classification system. It does not provide sufficient details for someone to reproduce the work. There are significant issues with the presentation of figures, tables, and referencing.
Pass (16-18)	The report offers a basic description of the classification system, but it lacks clarity and coherence. There are areas that could be more detailed. Reproduction of the work is possible with effort. LaTeX is used, but there are some issues with figures, tables, or referencing.
Merit (19-21)	The report effectively describes the classification system with clarity and organization. The details provided make it reasonably easy to reproduce the work. LaTeX is used appropriately, but there may be some minor issues with figures, tables, or referencing.

Score	Description
Distinction (22-24)	The report exceeds expectations in clarity and detail, making it easy to reproduce the work. The document has well-designed figures, tables, and accurate referencing.
Distinction (25-27)	The report is of exceptional quality, providing a clear and highly detailed description of the classification system. Reproducing the work is straightforward. The document has well-designed figures, tables, and accurate referencing.
Exceptional (28-30)	The report is flawless. It excels in clarity, detail, and reproducibility. The document has well-designed figures, tables, and accurate referencing. The document is of a publishable quality.

6.3. Performance (10/60)

Marks will be awarded based on the following rubric:

Score	Description
Fail (0-4)	The system does not run
Pass (5)	The system runs and produces results.
Merit (6)	The system runs and produces results that are better than the baseline system.
Distinction (7)	The system runs and produces results that are better than the baseline system and you have clearly documented some results showing hyper-parameter tuning.
Distinction (8)	The system runs and produces results that are better than the baseline system and you have thoroughly investigated the effect of hyper-parameter tuning.
Exceptional (9 or 10)	As above but with bonus marks for exceptional performance (e.g., top 2 or 3 scores in the class) and/or particularly thorough experimentation.

7. Submission

You will need to submit the following files:

- `train.py` - the Python script that trains your classifier on the training data and saves the trained model to a joblib file.
- `report.pdf` - a PDF report that describes your system and the experiments you have carried out. It should end with a table that reports the accuracy of your system on the evaluation data for each of the three sub-image sizes.
- `model.90.joblib` - a joblib file containing your trained model for the 90-pixel sub-images.
- `model.50.joblib` - a joblib file containing your trained model for the 50-pixel sub-images.
- `model.30.joblib` - a joblib file containing your trained model for the 30-pixel sub-images.

The assignment is due by 15:00 on Friday, 15th December. Standard lateness penalties will be applied.

⚠️ Academic Misconduct

The University takes academic misconduct very seriously. Academic misconduct includes plagiarism, collusion and fabrication of data. You should read the University's policy on academic misconduct to ensure you understand what is and what is not acceptable. The policy is available from https://www.sheffield.ac.uk/new-students/unfair-means. In particular, this is an individual assignment and any suspected collusion will be investigated. Do not share your code with anyone else.

Copyright © 2023 Jon Barker, University of Sheffield. All rights reserved.