Coursework 1 – Data Preparation and Classification

Date Released: Friday, 21st October 2022

Date Due: Friday 18th November 2022

This coursework is worth 20% of your overall mark

1. Introduction

In this coursework, you are required to mini research project in Machine Learning. You can choose to tackle a problem in either:

1. Computer Vision: Traffic Sign Recognition
2. Natural Language Processing: Sentiment Analysis

You may attempt both problems if you wish, but **please only submit one**. For each problem, the data will need to be prepared, feature and label arrays will need to created for training and testing, and models will need to be trained and evaluated.

2. Option 1: Traffic Sign Recognition from Images

This problem involves the automatic classification of images of speed signs. Your task in this project is to study the dataset, prepare it for machine learning, and to select the best classification model for automatically **determining the speed shown in the image**. This is a classification task and will require a **supervised learning** approach.

2.1. Dataset

The dataset for this problem is contained in the TrafficSignData.zip file. This zip file contains 2 folders:

1. The “Train” folder contains the images that will be used to train your model.
2. The “Test” folder contains the images that will be used to test and evaluate your model.

Both of these folders contain 4 folders with the names “40”, “80”, “90”, “100”, which contain the images showing speed limit signs for 40, 80, 90 100. The images are colour and 64x64 pixels in size.

2.2. Data Preparation

In order to use the images in your method efficiently, you will need to import each image into MATLAB, convert it to a double data type once imported, convert it to grayscale, and stack the columns to form a vector. I.e. your vector will take the form:

[column 1; column 2; column 3; …].

This will then need to be transposed to form a row vector.

2.3. Features and Labels

You will need to create one feature matrix for **training** with m_1 rows and n columns and one label vector for training with m_1 rows, where:

- m_1 is the number of training images, i.e. you have one image per row
- n is the number of pixels the images, i.e. $n = 64 \times 64$

You will need to create one feature matrix for **testing** with m_2 rows and n columns and one label vector for training with m_2 rows, where:

- m_2 is the number of testing images
- n is the number of pixels the images, i.e. $n = 64 \times 64$

2.4. Model Training and Evaluation

Please see the Model Training and Evaluation section below.

3. Option 2: Sentiment Analysis from Text

This problem involves the automatic classification of sentiment from recorded tweets from Twitter. Your task in this project is to study the dataset, prepare it for machine learning, and to select the best classification model for automatically **determining the sentiment displayed by the tweet**. This is a classification task and will require a supervised learning approach.

Please note that earlier versions of this assignment include an unfiltered publicly-accessible dataset called "text_emotion_data.csv". If you prefer to use the filtered version, you can use the version uploaded on 28th October 2022; if you use this version, the text file is called "text_emotion_data_filtered.csv".

3.1. Dataset

The dataset for this problem is contained in the SentimentAnalysisFata.zip file. This zip file contains 1 csv file: either "text_emotion_data.csv" or "text_emotion_data_unfiltered.csv" depending on which version you have. The csv file contains two columns:

1. "sentiment": this column lists one of four sentiments ("relief", "happiness", "surprise", "enthusiasm") that have been assigned to the corresponding tweet.
2. "Content": this column contains the text of the tweet

There are 8651 entries in this file.

3.2. Data Preparation

In order to use the tweets and their labels, you will need to import the csv file into MATLAB as a table, build a Bag of Words containing all of the tokenised tweets, remove stop words, remove any words with fewer than 100 occurrences in the bag (you can also try varying this number but it is not essential for this task), and build the full Term Frequency-Inverse Document Frequency matrix (tf-idf) for the resulting bag. You will also need to build a corresponding label vector from the column of sentiments.

3.3. Features and Labels

You will need to create one feature matrix for training by selecting the first m rows of the tf-idf matrix and all columns. You will also need to create a corresponding label vector with the first m labels. If you are using the unfiltered dataset, $m = 6921$; if you are using the filtered version, $m = 6432$.

You will need to create one feature matrix for testing by selecting all rows of the tf-idf matrix after row m (i.e. the remaining rows). You will also need to create a corresponding label vector.

3.4. Model Training and Evaluation

Please see the Model Training and Evaluation section below.

4. Model Training and Evaluation

4.1. Model Training

You will need to train and compare **any three** classification algorithms implemented in MATLAB. It is sufficient for this to use their default parameters but you are welcome to optimise these parameters to improve performance.

Table 1 shows some examples of classification algorithms in MATLAB and their function names as implemented in MATLAB's Statistics and Machine Learning Toolbox™. You may find it useful to have a look at this MATLAB documentation page on Supervised Learning Workflow and Algorithms for some ideas on how to structure your solution and select your algorithms. You are encouraged to make use of the suggestions on this page but feel free to explore further.

Table 1: MATLAB Classification of Algorithms and their function Names

Algorithm	**Function**
K-Nearest Neighbour	fitcknn()
SVM for Multiclass	fitcecoc()
Decision Tree	fitctree()
Naïve Bayes	fitcnb()
Discriminant Analysis	fitcdiscr()
Ensembles	fitcensemble()

To train a machine learning model, you need to use the syntax:

```
>> model = function(features,label)
>> predictions = predict(model,features)
```

For example, you might try:

```
>> knnmodel = fitcknn(training_features,training_label)
>> predictions = predict(knnmodel,testing_features)
```

4.2. Evaluation

Accuracy is measured by comparing the predictions from your models to the test labels in the dataset. It will be sufficient to calculate the accuracy as

$$accuracy = \frac{number\ of\ correct\ predictions}{total\ number\ of\ labels}$$

You should also investigate the results more thoroughly by analysing the resulting confusion matrix. You can obtain this with the `confusionchart` function. Type `help confusionchart` in MATLAB if you need help with this.

5. Submission

Please submit the following on Moodle on or before the **deadline of 4pm on Friday, 18th November, 2022**.

Your submission should be in the form of a zip file containing:

1. Your MATLAB your code written as a script (.m file), which is
    a. running without any errors
    b. Structured
    c. Including documentation/ detailed comments
2. A short 2 page report with the following sub-headings, that you would typically expect to find in a research report:
    a. Introduction
        i. This should be a paragraph, describing the problem that you are solving and potential uses for it.
    b. Data and Preparation
        i. Describe the content of the dataset
        ii. Describe the data preparation method
    c. Methodology
        i. Describe the model training and evaluation methods used, including your reasons for choosing the models. You should briefly describe each method but you do not need to give a detailed technical description.
    d. Results
        i. This should include presentation of your results, error analysis and observations
    e. Conclusion
        i. In this section, you should identify the model that you would recommend for use with your problem and justify your reasoning.

6. Marking

Marks and feedback will be returned by 9th December 2022. There are a total of 20 marks for this coursework and the coursework mark constitutes 20% of your overall mark for this module.

- 10 marks are allocated to your code, including correct implementation, appropriate structure and suitable annotation/documentation of your code using comments.
- 10 marks are allocated to your report, with 2 marks per section mentioned above.