CSC345/M45 Big Data and Machine Learning Coursework: Object Recognition

Policy

1. To be completed by students working individually.
2. Feedback: individual feedback is given on Blackboard within two weeks of deadline.
3. Learning outcome: The tasks in this assignment are based on both your practical work in the lab sessions and your understanding of the theories and methods. Thus, through this coursework, you are expected to demonstrate both practical skills and theoretical knowledge that you have learned through this module. You also learn to formally present your understandings through technical writing. It is an opportunity to apply analytical and critical thinking, as well as practical implementation.
4. Unfair practice: This work is to be attempted individually. You may get help from your lecturer, academic tutor and lab tutor, but you may not collaborate with your peers. **Copy and paste from the internet is not allowed**. **Using external code without proper referencing is also considered as breaching academic integrity.**
5. Submission deadline: The report and your implementation code in Python need to be submitted electronically to Blackboard by **11AM Monday 10 December**.

1. Task

The amount of image data is growing exponentially, due in part to convenient and cheap camera equipment. Teaching computers to recognise objects within a scene has tremendous application prospects, with applications ranging from medical diagnostics to Snapchat filters. Object recognition problems have been studied for years in machine learning and computer vision fields, however it is still a challenging and open problem for both academic and industry researchers. The following task is hopefully your first small step on this interesting question within machine learning.

You are provided with a small image dataset, where there are 10 different categories of objects, each of which has 1000 images for training and 100 images for testing. Individual image only contains one object. The task is to apply supervised learning algorithms to classify the testing images into object categories. The code to compute image features and visualize the image is provided. You can use it to visualize the images, compute features, transform them if necessary, e.g. PCA and LDA. You will then perform supervised classification and report quantitative results; writing this up into a 4-page report. You don't have to use all the provided code or methods discussed in the labs so far. You may add additional steps to the process if you wish. You are encouraged to use the implemented methodology from established Python packages taught in the last task of each labsheet (i.e. sklearn, skimage, keras, scipy…).

2. Image Dataset – Subset of CIFAR-10

We select a sub-set of 10 object categories from the complete CIFAR-10 dataset. Each category contains 1000 training images and 100 testing images, which are stored in two 4D arrays. The corresponding category labels are also provided. The size of each image is fixed at 32x32x3, corresponding to height, width and colour channel respectively. The training images will be used to train your model(s), and the testing images will be used to evaluate your model(s). You can download the image dataset and relevant code for visualization and feature extraction from the following URL: http://csvision.swansea.ac.uk/BDML/CW.zip

There are four NumPy files in the zipped directory, as follows:

- *trnImage*, 32x32x3x10000 matrix, training images (RGB image)
- *trnLabel,* 10000x1 matrix, training labels (1-10)
- *tstImage*, 32x32x3x1000 matrix, testing images (RGB image)
- *tstLabel,* 1000x1 matrix, testing labels (1-10)

The data is stored within a 4D matrix, and for many of you this will be the first time seeing a high dimensionality matrix. Although this can seem intimidating, it is relatively straightforward. The first dimension is the height of the image, the second dimension is the width, the third dimension is the colour channels (RGB), and the fourth dimension is the samples. Indexing into the matrix is similar to as with any other numeric matrix in Python, but now we deal with the additional dimensions. So, in a 4D matrix 'X', to index all pixels in all channels of the 5th image, we use the index notation X[:, :, :, 4]. So in a generic form, if we want to index into the i,j,k,lth element of X we use X[i, j, k, l].

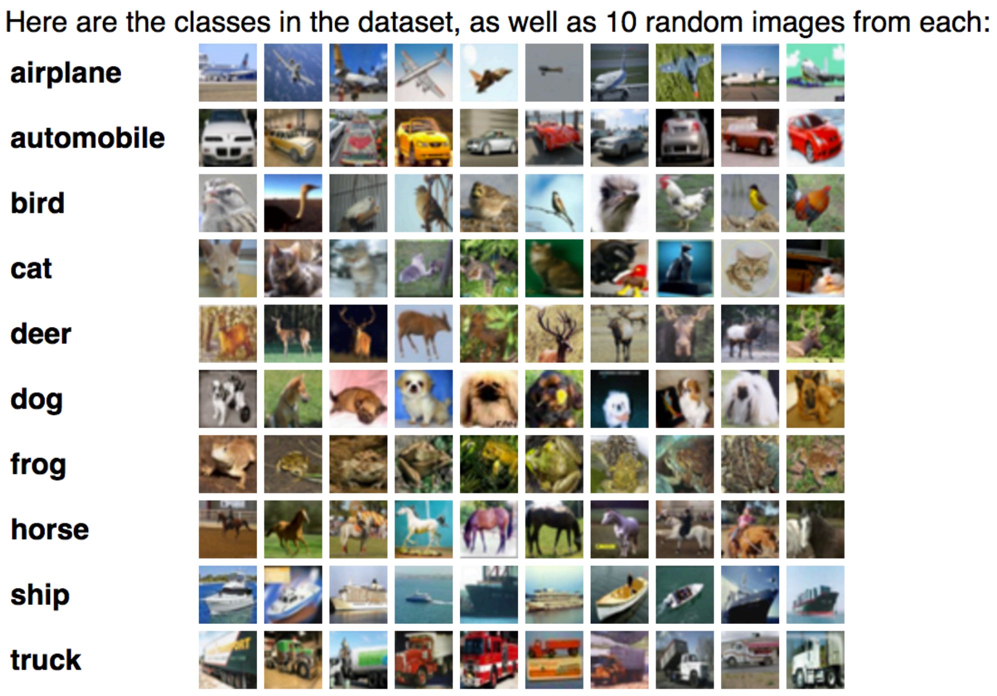
Here are the classes in the dataset, as well as 10 random images from each:

Figure 1. 10 Categories of CIFAR-10 Dataset

3. Computing Features and Visualizing Images

The function *computeFeatures* is provided to allow you to compute image features. A Jupyter notebook is provided to showcase how to use this function to obtain features we wish to train our models on, how to visualize these features as an image, and how to visualize a raw image from the 4D array.

The *computeFeatures* function utilises the Histogram of Orientated Gradients method to represent image domain features as a vector. You are NOT asked to understand how these features are extracted from the images, but feel free to explore the algorithm, underlying code and the respective Python package APIs. You can simply treat the features as the same as the features you loaded from Fisher dataset in the Lab work. You may need a *for* loop to compute the image features for each training and testing images.

4. Learning Algorithms

You can find all relative learning algorithms in the lab sheets and lecture notes. You can use the following algorithms (Python (and associated packages) built-in functions) to analyse the data and carry out the classification task. Please note that if you feed the learning algorithm with a large chunk of data, it may take a long time to train.

- K-Mean: Lab sheet 2
- Gaussian Mixture Model: Lab sheet 2
- Principal Component Analysis: Lab sheet 3
- Linear Discriminative Analysis: Lab sheet 3
- Support Vector Machine: Lab sheet 4
- Neural Networks: Lab sheet 4

5. Benchmark and Discussion

Your proposed method should be trained on the training set alone, and then evaluated on the testing set. To evaluate, you should count, for each category, the percentage of correct recognition (i.e. classification), and report the confusion matrix. The benchmark to compare against is 44.68%, averaged across all 10 categories. Note, this is only a reference, not a target.

Report

You are required to write a 4-page report to summarize your proposed method and the results. Your report should contain the following sections:

1. **Introduction**. Provide an overview of the problem, your proposed solution, and your experimental results. [10%]
2. **Method.** Present your proposed method in detail. This should cover how the features are extracted, any feature processing you use (e.g. clustering and histogram generation, dimensionality reduction), which classifier(s) is/are used, and how they are trained and tested. This section may contain multiple sub-sections. [50%]
3. **Results.** Present your experimental results in this section. Explain the evaluation metric(s) you use and present the quantitative results (including the confusion matrix). If you have tried multiple solutions, present all the results. [30%]
4. **Conclusion.** Provide a summary for your method and the results. Also, provide your critical analysis; that is the shortcomings of your method and how they may be improved. [10%]
5. **References.** Include references where appropriate. References are included in the page limit.

Page Limit: The report should be no more than **4 pages**. Font size should be no smaller than 10, and the text area is approximately 9.5x6 inches. You may use images but do so with care; do not use images to fill up the pages. You may use an additional cover sheet, which has your name and student number. Reports that exceed the specified page limit will result in penalties: 10% deduction for every over-length page.

Source Code: Submit your Python source code to Blackboard within a **single Jupyter notebook**. Remember to carefully structure and comment your implementation for clarity. Submit the code, together with your report, in **a single Zip file with the filename of 123456.zip**, where 123456 is your student number.

Assessment

The percentages listed above indicate the distributions of marks. This assignment is worth 20% of the total module credit. Submitted work without an implementation will lose the entire mark for section 4 and portions of marks for other sections.

Submission

Submit your work electronically to Blackboard. **Your report should be in PDF format only**. Compress your Python source code and report into **a Single Zip file**. The deadline for this coursework is **11AM Monday 10 December**.