COM2028 *Introduction to AI*
Assessment Overview and Coursework Guideline - Extended version

May 4, 2020

Overall Coursework Project Description

This coursework concerns the automated recognition of letters and words with the help of AI techniques. Letters and words can be given in various formats. The description and requirements have been revised to accommodate the changing assessment pattern due to Covid-19.

The original overall assessment pattern for COM2028 is shown in the following table, which shows the exam counts 50% of the module. This unit of assessment will now be in the form of the coursework. So the overall coursework will counts 80% of the module. This document provides the revision of the coursework requirements in order to meet the overall learning objectives for the module.

Unit(s) of Assessment	*Weighting Towards Module Mark (%)*
Coursework (individual)	30%
Exam (Two Hour Unseen Examination)	50%
Lab exercises and questions	20%
Qualifying condition(s) A weighted aggregate mark of 40% is required to pass the module	

The detailed breakdown of the new assessment patterns are:

1) Lab exercises and questions through lab tests (week 1-9): 20%

2) Coursework: a) Lab Tasks specified in lab sheets (ipynb files) in week 7-9: 15%; and b) Coursework Project: 65%

Details are provided as below:

Schedule

Week 1-9: Between week 1 and 9, there are lab exercises each week and questions (tests) that students should complete and submit through `SurreyLearn`. Those exercises and questions are mostly self-explanatory. On completion of each lab work, preliminary scores will be obtained through answering questions on `SurreyLearn`. Key solutions to the lab exercises will be available one week after each lab session. The scores on all lab tests will count for 20% of the module under the Unit of Assessment ”Lab exercises and questions”. Lab 7-9 (SVMs, GAs, and Baysian classifier on spam filtering) will count further 15% of the module as part of the extended coursework. For lab in week 7-9, apart from completing lab tests, please also submit completed lab Tasks specified in these labsheets (in *.ipynb files), alongside the deliverables for the coursework project in week 14, Monday at 4pm.

Regular engagement with the lab tasks will enable good digestion of the module content and prepare students for the coursework project. Coursework guidelines will be provided before the end of week 5. As the course is progressing, we will start to prepare for the coursework, getting familiar with the related concepts and techniques.

Week 7-13: You will be focusing more on the coursework project. You may find that the foundation built upon all lab activities is crucial for a successful completion of the coursework. Most importantly, with such foundation, working through the challenging tasks in the coursework becomes more enjoyable and satisfying.

Week 10-13: There will be 10 sessions of tutorial meetings between the teaching team (Dr Lilian Tang, and Dr Yongxin Yang) and a group students (around 10 to 20 students each time), to discuss students' progress and clarify any queries. A set of questions on the subject area will be circulated for students to prepare for the meeting. Students should also try to complete all the lab work. The schedule of the meetings will be sent to all students through Microsoft Teams. Please download Microsoft Teams and login using University account (it should work on both computer and phones). If you cannot attend the scheduled slot, please feel free to attend a different one that suits you better.

Week 14, Monday, 8 June 2020 at 4pm: Coursework due. The following deliverables are required to be submitted to `Surreylearn`.: 1) a poster on the coursework project, 2) the developed programs, and 3) completed Tasks in lab 7-9.

Coursework Project

Coursework Objectives

- To understand the workflow of developing a useful computer vision system
- To develop machine learning techniques and evaluate their performance

Coursework poster

It is important to demonstrate your understanding of the subject area. There are many useful guidelines on how to design an academic poster, for examples: `http://www.surrey.ac.uk/library/documents/learning/SPLASH%205_Poster%20Design.pdf`, or `http://guides.nyu.edu/posters`

The length of the poster can be up to 3 pages of A0 size. Additional screenshots of the software outputs could be added at the end of the poster as an appendix. Please note everything will be in digital format, no hard print copy is required.

Source code

Any developed source code should be submitted.

Preparation of the deliverables: Students should consider including sufficient evidence of your work through the poster, and the program code covering the following areas in the marking criteria.

Marking criteria

- The **introduction** to the project, providing appropriate background information, and detailing the objectives of the project. Literature review should be carried out on relevant topics and how they may relate to your work. You are expected to research and discuss other sources of information, but must show their origins by referencing all sources used. The reference list should be included in the poster. (10/100)

- **Analysis** of the problems, research and technical issues, challenges, and description of the approach you have developed with justification. Effective graphic illustration will be appreciated. Multiple classification techniques should be designed and implemented, including: K-nearest Neighbour (KNN), multilayer perceptron neural networks (fully connected neural networks), convolutional neural networks (CNNs) and Support Vector Machines (SVMs)). You don't have to implement all classifiers for each task in the project, but aim to cover all these techniques in order to demonstrate your mastery of them. Sometimes it is useful to have multiple techniques for the same task so you can make a comparison. You may find you can use the same code for different task with minimum change. (35/100)
- **Implementation and evaluation** of the methods. You need to discuss the system results, present the data you used, how many of them for training and testing and what is the recognition accuracy etc. Figures and tables can be useful to present such information, especially when multiple approaches are developed and comparison of various approaches/choices are conducted. ROC curve, confusion matrix, learning curve, error curve etc will be good graphics for presentation. (35/100)
- Discussions and findings of your investigation/study . (10/100)
- Clarity in presentation. (10/100)

You should try to answer the following questions in the poster,

- What is the state of art in this field? Is my work making a new contribution to the field or it is just re-inventing the wheel? (both have their great value as we are learning in the field.)
- Do I understand how each technique works in depth?
- What are the advantages and weaknesses of these techniques?
- Is the comparison vigorous? has each technique been developed to its optimal level for a fair comparison and evaluation?
- If the system fails to achieve the desired project goals, can I explain why? What might be the issues in various stages (e.g. data preparation, feature extraction, training) that give rise to a poorer per-

formance? Is the training appropriate and sufficient, or whether the training is overfit/underfit? is the evaluation of the developed system thorough enough? is the developed system scalable or generalisable?

- What is your recommendation?

Coursework Project Description

Recognising printed pages for multiple languages

The first part is to extract letters and words from an image of a printed page. So the system will take an image of a printed text, and output a text of the words in that image. There are so many factors that make this task much easier than recognising the words from a handwritten note. The first two steps for this task is to *build a dataset* that is suited for training, and then *train a letter recogniser* based on such dataset. For the training part, you shall try to use the methods that we have already practised in the lab (SVM, K-NN, neural network, and CNN etc) or a combination of them.

Prepare data from at least two kinds of languages, such as English and Chinese. Develop a system that can recognise both types of languages.The software for each language could be the same, but they will be trained using different sets of data for individual language.

As shown in the sample data provided under Coursework folder on SurreyLearn, we can generate our own training samples by repeating the basic letters in different fonts, and automatically extract individual letters as training samples to feed into the classifiers. Sample code "extract_letter.py, sorting_charactor.py, or sorting_characterUpdate.py" are similar with different levels of hint. You can use one of these as a starting point. But feel free to develop your own approach. There are open source data that you can also use.

Handwritten notes

The second task is to create a similar engine, but this time the engine should work on handwritten notes. Again, your task is to create the training dataset, using your own handwritten notes or those you can find from open source. Note the data should be pages of handwritten notes, then train a system to recognise each letter in the notes. Experiment with different classifiers, and report your findings, comparing with the recogniser that works on printed pages.

Signature recognition

Using similar techniques, build a system that accepts as an input an image of a signature, and returns the name of the person that this signature belongs to. Of course, you will have to train the system, using different people's signature data. You can start with only two people, but it would be worth to add more signature data for each person, and see how well your system can deal with that. The more training data you have, the better the system should learn. Of course, you need to make sure the testing data are not included in the training. Again, you can choose the type of classifier as well as the kind of features to use, you can experiment in order to find out the best choice (and report on why you have selected this specific configuration).

You may find it easier to treat the whole signature as one piece of image rather than trying to separate each letter (part) out of the signature.

NOTE: Please make sure you have asked for permission if you are collecting signatures from other people so they are happy for you to use them in your study. You may find some public data from open source.

Testing - Evaluation

For the testing stage, a simple comparison on the percentage of correctly classified letters would be enough. You can even go to a higher level, and incorporate a word analysis facility into the system. However, by doing that, it may mean that you also have to add text related capabilities to your system (for example a simple word or grammar checker etc).

Try to use tables, ROC curves and confusion matrices etc to illustrate your evaluation results.

Tips

- Note that in order for the labeling and the letter segmentation to work well, your handwriting should be non-continuous! Do not stick the letters and the words together, it will be much more difficult to segment a word that has no spaces between the letters. For those interested, it is still possible to segment such words and understand the letters, but it involves sliding windows and averaging responses, something that is out of the scope of this coursework.

- Obviously for both cases, the easiest way to create the training data is to fill out pages with single letters. This will make it easy to program the whole system. For example if you want to train the system to recognise the letter *a* just fill out a page with a number of *a*'s, lets say 100, and then pass these 100 objects to your recogniser together with the label that represents the letter *a*. You can create data in a similar fashion for the second language.

- Multiple fonts. You can make your system more robust, by using multiple different fonts in the training process (of course for the task on recognising printed page).

- Some training and testing samples for print English text recognition are provided in `WritingRecognition.zip`. **These are only sample data and sample code You can use these data as a starting point. You should modify the code as appropriate, or come up with your own code, and extend the training and testing sets as needed.**

- The program `extract_letters.py` is given, and it has a basic method of segmenting a page and returning the recognised letters. And more hint was given in "sorting_charactor.py, or sorting_characterUpdate.py". As it potentially can segment individual letters from rows of text, these code is also useful when you try to recognise the whole pages of text during the prediction phase. The size of bounding boxes may need to altered if different fonts, language are used. If you use HoG features, then you have to consider the appropriate size of cells, blocks etc. in HoG parameters.

- It is always better to visualise the results, for example, you can use `matplotlib` to plot graphs on the success rates, and the recognition rates etc.

Image Segmentation

If you take a look at the code in the `extract_letters.py` program, you will see that what this program does is to create bounding boxes for all the letters in a scanned image of a document.

By using the variable `rect`, you are able to get the bounding box for each of the letters. As we have seen before, you can use `numpy`'s slicing abilities, in order to get a part of a bigger matrix. For example if the variable `image` is a matrix `[100,100]` and you only need a small part that is `[10,10]` in the top left part of the image, you can write

`part = image[0:9,0:9]`

The syntax `list[x:y]` means returning the part of the list that is between `x` and `y`.

So you can use exactly the same logic, in order to extract each letter from the original image using the bounding box.

scipy.imresize

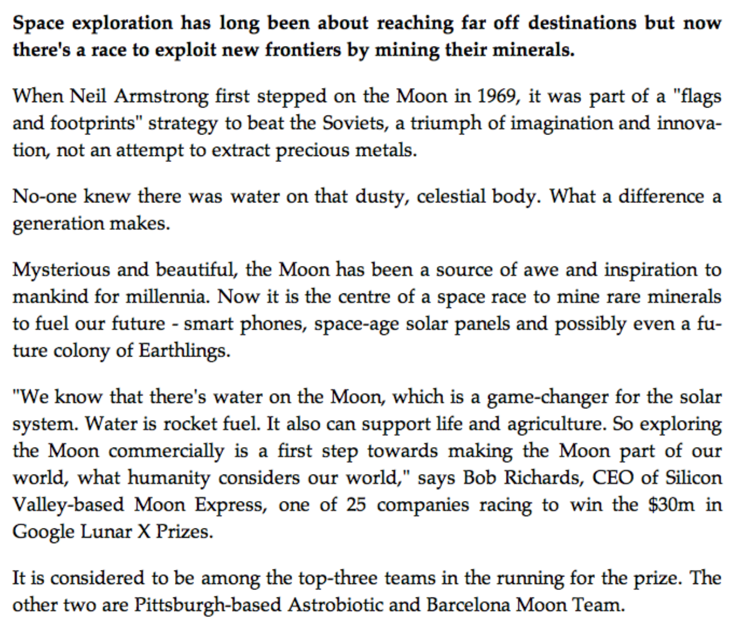
Letters have different shapes and sizes. So it it better if you normalise everything for easier classification process. A logical size should be around `[20,20]`.

```
letter_raw = image[bounding_box[0],etc....]
letter_norm = imresize(letter_raw,(20,20))
```

Letter extraction and size normalisation

Now you can use `letter_norm` both for training purposes and testing purposes. You *have* to keep everything constant, during the training and the testing process. For example, don't train your system with `[20,20]` patches and then try to run it on `[20,15]` patches.

Example of segment individual letters from an image



Space exploration has long been about reaching far off destinations but now there's a race to exploit new frontiers by mining their minerals.

When Neil Armstrong first stepped on the Moon in 1969, it was part of a "flags and footprints" strategy to beat the Soviets, a triumph of imagination and innovation, not an attempt to extract precious metals.

No-one knew there was water on that dusty, celestial body. What a difference a generation makes.

Mysterious and beautiful, the Moon has been a source of awe and inspiration to mankind for millennia. Now it is the centre of a space race to mine rare minerals to fuel our future - smart phones, space-age solar panels and possibly even a future colony of Earthlings.

"We know that there's water on the Moon, which is a game-changer for the solar system. Water is rocket fuel. It also can support life and agriculture. So exploring the Moon commercially is a first step towards making the Moon part of our world, what humanity considers our world," says Bob Richards, CEO of Silicon Valley-based Moon Express, one of 25 companies racing to win the $30m in Google Lunar X Prizes.

It is considered to be among the top-three teams in the running for the prize. The other two are Pittsburgh-based Astrobiotic and Barcelona Moon Team.

Figure 1: The original image that contains a chunk of text

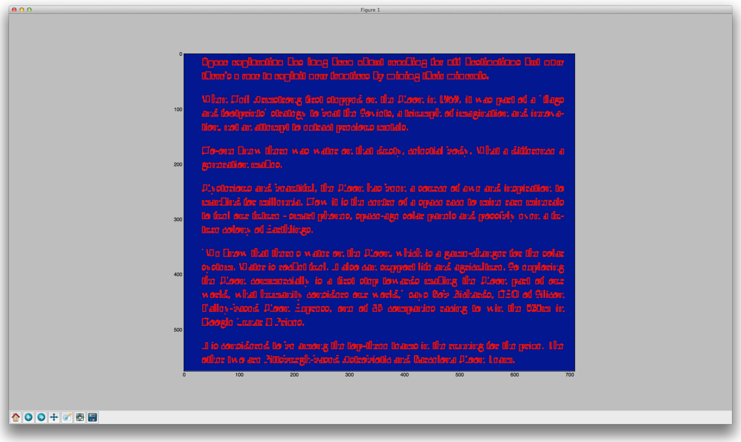
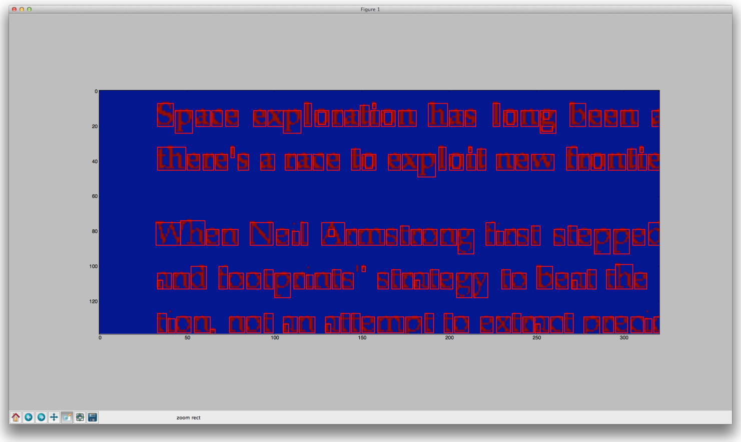


Figure 2: Extracting all the elements from the image. Similar to what we did in Week 2 with the coins, but for much more elements. *(zoom in to see clearer details)*



Space exploration has long been

there's a race to exploit new frontie

When Neil Armstrong first steppe

and footprints'' strategy to beat the

tion, not an attempt to extract preci

Figure 3: The first part from the image above *(zoom in to see clearer details)*