# COMP3702/COMP7702

# Assignment 2: Handwritten letter recognition

Due 5pm, Friday, 25 October 2013 (extension with no penalty to 5pm Mon 4th Nov).
**This assignment counts 20% toward the final marks.**

## Purpose

This assignment is intended to practically acquaint you with machine learning methodology and techniques. Additionally, it should provide some understanding of how simple pattern recognition tasks, such as handwritten letter recognition (popular in handheld and ubiquitous computing, and optical character recognition applications), can be implemented and how you as the system designer can visualise aspects of the components as they are learning and/or the final system design.

## Files

zipped source code; jar package; zipped data set; report template; marking sheet.

## Links

package description; Eclipse project creation tutorial.

## Code and Data

The supporting material (zipped source code, jar package, package description, zipped data set) can be used freely in your work. The supporting code is in Java and you should use and modify this to complete the assignment. If you do not know Java currently, you will not need to learn a great deal of Java to do the assignment, but working in a pair might be a good idea.

The data set of over 4000 characters was generated by students in a previous year using a version GenerateLetters.java with only letters used. The working code may give you ideas for pre-processing (for Part B).

Note that letters created using GenerateLetters.java have their rows and columns swapped, likely because BitmapPanel.java does everything in terms of (x,y) positions and Bitmap.java does everything in terms of row, column (ie: y,x). All of the letters in the data set have this format. All of the letters in the test set will have this format. It may be important to be aware of this setup if doing any pre-processing.

Also note that some of the letters in the data set are not recognisable as the letters that they are supposed to be (e.g. lines 3208-3293). If you think that some of the example letters are so poor that they are negatively affecting learning, you could leave them out of your dataset. However, you should explain and justify this, including consideration of any risks of doing so.

In describing your results, you need to provide visualisations that demonstrate how the system is changing during learning and/or how the final performance is achieved. You will need to add commands to the codebase to export the data required for visualising. Use csv format for exporting the data. The graphs can be created using any program (eg excel, matlab).

## Administration

The assignment is worth 20% of your final grade and you may work individually or in pairs. Working individually will attract an automatic 1 bonus mark.

The assignment is due 5pm, Friday 25th October 2013. An extension will be automatically granted to 5pm Monday 4th November, but there will be little or no help available during swotvac.

Final submissions received before 5pm 25th Oct will receive an automatic 1 bonus mark.

There are two sections in this assignment. Section A is worth 10 marks. In Section B, there are more than the maximum marks available; you may attempt all of the 16 marks available, but your mark for this section will be capped at 10 marks. Your grade will be capped at 20 marks for this assignment, even if your raw score is over 20.

You are required to submit three separate components for this assignment, these are:

1. A single file containing all the source code files (e.g. *.java), including any you modify. Modified sections of source code should include comments and overall, the code should compile. There is no excuse for handling in a program which does not compile/interpret or run. If you are having trouble, some features may be omitted. Package and submit all the source code (*.java), including unmodified files, maintaining directory structure as necessary. You may package the source code into e.g. a .zip or .jar file.
2. Submit your best classifier, named Classifier_XX.java (where XX is your student number(s)) and the serialized instance of this class, named classifier_XX.ser (the file you get when you use bitmap.Classifier.save). All additional java classes should be physically contained within Classifier_XX.java as inner/private classes. Your class must belong to the bitmap package. Moreover, your class must extend (i.e. be a subclass of) the original bitmap.LetterClassifier (if not, your serialized classifier file will not load). You can assume that you have access to all java classes and data files that are part of the distributed code. Your program should not overload or rewrite existing functions as the test script may not work. Make sure that the serialized file is generated by the same class definition as you submit; even changing the class name will invalidate the serialized file. The test system will re-construct your classifier instance. The classifier should operate very quickly (less than 1 second on a standard PC) and load within 3 seconds (on a standard PC). From a letter recognition point of view, the test application will work precisely as bitmap.UseClassifier.java.
3. A pdf with the written component of the assignment (see the template for general guidance). Note that the documentation differs from previous years in requiring a visualisation for each learner: the neural network weights during learning, and the information gain in the decision tree. These visualisations are pass/fail (no visualisation means no marks for that section). Indicate in your pdf which source code files you modified and if there are any new files. Also indicate in your pdf exactly what parts you have attempted, in part or in full, so that we know what to look for when marking. In the report, describe how to compile/interpret and run your program (include name and version of compiler/interpreter).

Submission of all parts is via Blackboard. Use your student number(s) within the pdf and code, and as part of the pdf and packaged code filenames.

If you resubmit your assignment, please resubmit all files relating to your solution. You may resubmit as many times as you like. Your most recent submission will be the only submission marked.

In line with University policy collusion and plagiarism will not be tolerated; see the course profile for more details on these topics.

You can discuss aspects of the assignment with other members of the class using the comp3702 Blackboard discussion forum. The forum is also monitored by the tutors and lecturer.

## The Assignment

You have been given a large dataset which contains samples of “handwritten” letters.

Source code has been provided which implement a simple version of ID3 decision trees and single-layer neural networks.

In this assignment, you are asked to investigate how machine learning techniques create models to classify instances of handwritten letters.

### *Marking*

The following is the marking scheme. There are two main sections and bonus marks. All parts of Section A should be completed, although you can choose which technique you will use for your advanced classifier. You can complete any number of parts in Section B, which is capped at 10 marks. You should carefully document all the parts of the assignment in a single PDF. Your programs should be well designed and should use meaningful variable names, correct indentation of code, useful comments, etc.

**Section A: Basic and Advanced Classifier (10 marks)**

You should complete all of the following Parts (1-4), choosing one technique for the advanced classifier (Part 3).

Divide the data set into at least two sets (training and test), adding extra data as necessary. Investigate and optimise the performance of the already implemented single layer neural network. Investigate and optimise the performance of the already implemented ID3 decision tree. Implement an advanced classifier using your chosen technique, and investigate and optimise its performance. Discuss the differences in performance obtained for the optimised single-layer neural network, the optimised ID3 decision tree, and your optimised advanced classifier.

Provide a visualisation (one example graph per classifier) demonstrating the weight changes during learning (for one node in neural networks) and the information gain (for each level in one path through decision trees).

| Part | Description | Marks |
|---|---|---|
| 1 | **Single-layer Neural Network**<br><br>Investigate and optimise the performance of the already implemented single layer neural network. Optimisations include the size of the | 2 |

| | | |
|---|---|---|
| | test/training sets, the learning rate, and the number of iterations of learning.<br><br>Your assignment must contain the following:<br>1. Describe the optimisations investigated (0.5 marks) and<br>2. Describe and discuss the performance of the system on training and test data for the optimisations investigated (1.5 marks).<br>3. Visualisation: i.e. graph of weight values vs time of the changing weights for the output node for the letter “A”. (pass/fail) | |
| 2 | **ID3 Decision Tree**<br><br>Investigate and optimise the performance of the already implemented ID3 decision tree. Optimisations include the size of the test/training sets, and the two thresholds used for deciding when to create a leaf node (proportion and number of samples).<br><br>Your assignment must contain the following:<br>1. Describe the optimisations investigated (0.5 marks) and<br>2. Describe and discuss the performance of the system on training and test data for the optimisations investigated (1.5 marks).<br>3. Visualisation: i.e. graph of information gain vs level in decision tree for a path from the root to one of the (interesting) leaves. (pass/fail) | 2 |
| 3 | **Advanced Classifier**<br><br>You should choose one machine learning technique (either improving on one of the provided machine learning techniques or implementing a different technique) to design and train an advanced classifier (see details below).<br><br>Your assignment must contain the following:<br>1. Describe your implementation or how the basic implementation was extended, including references to published literature (1.5 marks),<br>2. Provide correct implementation in Java, including in your report a description of the code or how the supplied code was altered and how your implementation should be run (1.5 marks), and<br>3. Describe and discuss the performance of the system on training and test data for the optimisations investigated (2 marks).<br>4. Visualisation of an aspect of the advanced classifier (pass/fail)<br>5. Code for your best classifier (see Admin section point 2 above). (pass/fail) | 5 |
| - | **Multi-layer Neural Network**<br><br>(See Russell and Norvig, p. 731-736.) | - |

|  |  |  |
|---|---|---|
|  | Modify the source code provided to extend the single-layer neural network to produce a more advanced classifier by adding hidden units to the network giving it (at least) two layers of weights. Optimise the performance of your multi-layer neural network; optimisations could include the number of hidden units, the learning rate, and the number of iterations of learning. |  |
| - | **ID3 Decision Tree with Pruning**<br>(See Russell and Norvig, p.705-706.)<br>Modify the source code provided to extend the decision tree to produce a more advanced classifier by using any relevance-based heuristic to remove paths ("prune" decisions) in the decision tree. Optimise the performance of your pruned decision tree; optimisations could include the cut-off used for pruning, and the two thresholds used for deciding when to create a leaf node. | - |
| - | **Different Machine Learning Technique**<br>Implement an advanced classifier with a different machine learning technique (not decision trees or neural networks); one option is a Naive Bayes' Classifier. Optimise the performance of your advanced classifier; optimisations available will depend on the technique that you use. | - |
| 4 | **Comparison**<br>Discuss the differences in performance obtained for the optimised single-layer neural network, the optimised ID3 decision tree, and your optimised advanced classifier.<br>Your assignment must contain the following:<br>1. Discuss the relative performance of the classifiers. Compare the optimised classifier for each technique, discussing the success of the classifiers on test sets, the time taken to create that classifier, and the relative success rates for different letters. (1 mark) | 1 |

**Section B: Refinements (capped at 10 marks)**

Any number (0 to 4) of the following parts (1-4) may be completed. All are assigned the same marks (4 marks each). Although 16 marks are available, this section will be capped at 10 marks.

For each task completed, your assignment must contain the following:

1. Describe your implementation of the method, including references to published literature (1 mark),
2. Provide correct implementation of the method (1 mark), and
3. Describe the performance on training and test data, especially in comparison to the performance without the task implemented (2 marks).
4. Include a visualisation of the learning components changing over time (neural networks), or the information gain at each level (in a decision tree), or choose another way of revealing how the system is learning. (pass/fail)

| Part | Description | Marks |
|---|---|---|
| 1 | **Pre-processing**<br>Pre-process each data instance before it is presented to the classifier for training/testing. Techniques include centring the letter in its 32 * 32 bitmap square, Gaussian blur each pixel with its neighbours, calculating the principal components of the training data (or another transform such as a wavelet transform), or compressing the letter into a smaller bitmap area (e.g. 8 * 8). Several techniques may be used together. | 4 |
| 2 | **Ensemble of Classifiers**<br>Make multiple copies of your classifier, and train these copies on subsets of your training data. Produce an overall prediction in response to a test pattern by combining the results from multiple classifiers via a voting system. See Russell and Norvig, section 18.10 on p. 748-752 for some background. | 4 |
| 3 | **Overtraining Prevention**<br>There are a number of methods that may be used to ensure the classifier is not overtraining, some of which are technique specific. One option is to partition your training set into two new sets (a training set, and a validation or verification set) and to use this validation set to estimate when to stop training the classifier. | 4 |
| 4 | **Parameter Optimisation or Other Refinement**<br>Implement some other technical refinement that aims to improve the performance of the classifier and that has not been specified. | 4 |
| **Bonus Marks (up to 2 bonus marks available)** | | |
| Part | Description | Marks |
| 1 | **Individual Work** | 1 |
| 2 | **Submission by 5pm 25th October.** | 1 |