

COMP3702/COMP7702

Assignment 2: Handwritten letter and number recognition

Due 5pm, Friday, 21 October 2011.

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/digit recognition (popular in handheld and ubiquitous computing, and optical character recognition applications), can be implemented.

Code and Data

The supporting material can be used freely in your work: [zipped source code](#), [jar package](#), [package description](#), [zipped data set](#).

The supporting code is in Java and we recommend that you to 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 5000 characters was generated by students in a previous year using a version GenerateLetters.java with only letters used. The numbers data set of 400 characters was generated using a variation of the current GenerateLetters.java with only numbers produced. You may choose to generate a larger data set using GenerateLetters.java, and the working code may also give you ideas (e.g. for pre-processing).

(See also [extra information](#))

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. In this assignment, there are more than the maximum marks available. You may attempt more than 20 marks (to create a “buffer” of marks), but your grade will be capped at 20 marks for this assignment, even if your raw score is over 20. You are required to submit two separate components for this assignment, these are:

- 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.

- A pdf with the written component of the assignment (see [template/example \[pdf\]](#) for general guidance). Please 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.

Submission of both parts is via the ITEE online submission website found at <http://submit.itee.uq.edu.au>. **Your group name should be the student number of the person submitting the assignment.** You should also use your student number(s) within the pdf and code, and as part of the pdf and packaged code filenames. **Note: There will be marks deducted if your group name does not correspond to a student number or if your PDF and code do not contain your names and Student IDs.**

If you resubmit your assignment, please resubmit all files relating to your solution. 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.

Each group must submit an assignment cover sheet signed by all members of the group. The online submission system will offer you a printable cover sheet as you submit your assignment. Otherwise, a suitable ITEE assignment cover sheet can be obtained from <http://studenthelp.itee.uq.edu.au/assignments/>. You should submit your completed form at your tutorial, at the lecture, or to 47-308 as soon as possible after submitting your assignment.

You can discuss aspects of the assignment with other members of the class on the comp3702 newsgroup (uq.itee.comp3702 or see my.uq.edu.au). If you want to ask the lecturer any questions, sending an email to ruth@itee.uq.edu.au is the best option.

The Assignment

You have been given a large dataset which contains samples of “handwritten” letters and a smaller dataset of “handwritten” numbers. You are to investigate how machine learning techniques create models to correctly classify instances of handwritten letters and numbers.

Source code has been provided which implement a simple version of ID3 decision trees and single-layer neural networks (see Code and Data above).

Part	Description	Marks
1	Comparison of Optimised Single-layer Neural Network and ID3 Decision Tree You should complete all of the following tasks (1A-1D) 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. Discuss the differences in performance obtained for the optimised single-layer neural network and the optimised ID3 decision tree.	
1A	<p>Data Sets</p> <p>Build a series of training and test data sets that can be used for parts 1B and 1C (and for later parts as needed). The data sets should contain both letters and numbers.</p> <p>Your assignment must contain the following:</p> <ol style="list-style-type: none"> 1. An overview of the content of each training and data set (numbers of examples, classes of examples), explaining how you obtained and sorted the data, and how combinations of the data sets will be used in parts 1B and 1C. (1 mark) 	1
1B	<p>Single-layer Neural Network</p> <p>Investigate and optimise the performance of the already implemented single layer neural network.</p> <p>Your assignment must contain the following:</p> <ol style="list-style-type: none"> 1. Describe the optimisations investigated (0.5 marks) and 2. Describe and discuss the performance of the system on training and test data for the optimisations investigated (1.5 marks). <p>Optimisations include the size of the test/training sets (as described in 1A), the learning rate, and the number of iterations of learning.</p>	2
1C	<p>ID3 Decision Tree</p> <p>Investigate and optimise the performance of the already implemented ID3 decision tree.</p> <p>Your assignment must contain the following:</p> <ol style="list-style-type: none"> 1. Describe the optimisations investigated (0.5 marks) and 2. Describe and discuss the performance of the system on training and test data for the optimisations investigated (1.5 marks). <p>Optimisations include the size of the test/training sets (as described in 1A), the two threshold used for deciding when to create a leaf node (proportion and number of samples).</p>	2
1D	<p>Comparison</p> <p>Discuss the differences in performance obtained for the optimised single-layer neural network and the optimised ID3 decision tree.</p> <p>Your assignment must contain the following:</p> <ol style="list-style-type: none"> 1. Discuss the relative performance of the neural network and decision tree, stating the optimisations for each. A more in 	1

	depth comparison should be made of the optimised neural network and decision tree, for example discussing the time taken to create that classifier, and the relative success rates for different letters and numbers. (1 mark)	
2	<p>Advanced Classifier</p> <p>You should complete one of the following tasks (2A-2C).</p> <p>Improve on one of the provided machine learning techniques or use a different technique to design and train an advanced classifier.</p> <p>Your assignment must contain the following:</p> <ol style="list-style-type: none"> 1. Describe your implementation or how the basic implementation was extended, including references to published literature (1 mark), 2. Provide and describe pseudo-code for your implementation or extensions to the basic implementation (0.5 marks), 3. 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 4. Describe and discuss the performance of the system on training and test data for the optimisations investigated (2 marks). 	5
2A	<p>Multi-layer Neural Network</p> <p>(See Russell and Norvig, p. 731-736.)</p> <p>Having completed Part 1B, you can now extend the single-layer neural network to produce a more advanced classifier. You are to modify the appropriate source code provided. You should optimise the performance of your advanced classifier.</p> <p>Add hidden units to the single-layer neural network to give it two layers of weights. Optimisations include the number of hidden units.</p>	-
2B	<p>ID3 Decision Tree with Pruning</p> <p>(See Russell and Norvig, p.705-706.)</p> <p>Having completed Part 1C, you can now extend the decision tree to produce a more advanced classifier. You are to modify the appropriate source code provided. You should optimise the performance of your advanced classifier.</p> <p>Use any relevance-based heuristic to remove paths (“prune” decisions) in the decision tree. Optimisations include the cut-off used for pruning.</p>	-
2C	<p>Different Machine Learning Technique</p> <p>Implement a classifier with a different machine learning technique</p>	-

	and optimise its performance (that is not decision trees or neural networks, for example a Naive Bayes' Classifier).	
3	<p>Refine study</p> <p>Any number (0 to 4) of the following tasks (3A-3D) may be completed. All are assigned the same marks (4 marks each, up to 16 marks available).</p> <p>For each task completed, your assignment must contain the following:</p> <ol style="list-style-type: none"> 1. Describe the method, including references to published literature (1 mark), 2. Describe your implementation (1 mark), 3. Provide correct implementation of the method (1 mark), and 4. Describe the performance on training and test data, especially in comparison to the performance without the task implemented (1 mark). 	–
3A	<p>Create an Ensemble of Classifiers</p> <p>Make multiple copies of your classifier, and train these copies on random 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.</p>	4
3B	<p>Implement a pre-processing technique</p> <p>Implement some pre-processing technique that is run on each data instance before it is presented to the classifier for training/testing and which you feel may help the classifier perform better. Ideas in this area may include centring the letter in its 32 * 32 bitmap square, emphasising curves or straight edges that appear in the instance, or even compressing the letter into a smaller bitmap area (e.g. 8 * 8). The choice here is up to you.</p>	4
3C	<p>Overtraining Prevention</p> <p>There are a number of techniques that may be used to ensure the classifier is not overtraining. 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. Again, the choice is up to you.</p>	4
3D	<p>Additional Refinement</p> <p>Implement some technical refinement that aims to improve the performance of the classifier and that has not been specified in 3A-C.</p>	4
4	<p>Competition</p> <p>Enter your trained classifier in a class competition, to be tested on a</p>	2

	<p>new set of data (details given below).</p> <p>+1 mark for entering the competition with a classifier that compiles, makes relevant use of a machine learning algorithm, and produces predictions for all test cases.</p> <p>+1 mark for the best four submissions from the competition. This will largely be determined by classifier accuracy on the test set, but other factors will be considered if necessary.</p>	
-	<p>References</p> <p>Published literature should be referenced throughout your report, with a reference list provided at the end of your report.</p> <p>Note that your references should include more than your textbook and that while Wikipedia is a good place to start looking it is not a good reference.</p>	1
-	<p>Individual Work</p>	1

Competition Details

You have the option of entering your classifier in a class competition, which will be run during the final lecture of the semester (Tuesday 25 October). Your classifier code will be tested on a new set of data to see the proportion of these examples which it correctly classifies. The results will also be placed on the course web page. This is not compulsory but is highly recommended (both for seeing the results of independent testing and for the possibility of extra marks).

To take part in the competition, submit your classifier code, 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`), before 9am on Monday 24 October. (Note that this is after the rest of the assignment is due.) Submission is via the ITEE online submission website found at <http://submit.itee.uq.edu.au> to the assignment competition folder.

All additional java classes should be physically contained within your java file (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. 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 competition system will re-construct your classifier instance. Then each new letter is tested on your classifier by calling `Classifier_XX.test(Bitmap map)`. The classifier needs to operate very quickly (much less than 1 second on a standard PC) and load within 3 seconds (on a standard PC). The program will be tested and the coordinator reserves the right to disqualify submissions (especially if they don't compile without modification). From a letter recognition point of view, the competition application will work precisely as `bitmap.UseClassifier.java`. In the competition your classifier will only be identified by the name you give it (in case you

wish to remain anonymous in class). **Please change the name of your classifier from the default “NN Classifier 1” or “ID3 Classifier 1”.**

Only one classifier per group can be used in the competition (but it can be an ensemble). To prepare for a successful competition you should try a number of different configurations, evaluate them objectively, and select one (or a combination) of them. Note that you must make relevant use of a machine learning algorithm. It is not sufficient to come up with a smart program.

Participating in the competition will attract 1 bonus mark (code compiles, meets criteria and produces predictions for all test cases). The best four submissions from the competition will receive another bonus mark. This will largely be determined by classifier accuracy on the test set, but other factors will be considered if necessary.

Java notes:

-Read the [package description](#) to understand the classes that you will need to change for this part.

-You may find an [Eclipse project creation tutorial](#) useful.