

Assignment 2

- 26 letters: A-Z, encoded 0-25.
- 5605 examples in cleaned dataset (~200 of each).
- Assume that each letter is equally likely (not true in English or other natural languages).
- Notation in code is older form, e.g. d for desired output, y for network output.
- Encoded using 1 network output per letter – correct output = 1, rest = 0.
- That's the desired outputs (targets).

bitmap

Class Summary	
Bitmap	A Bitmap holds a matrix of bits (true or false, on or off, 1 or 0).
ClassifiedBitmap	A subclass of Bitmap and extends it by adding a classification (a target class).
Classifier	A null model for classifiers acting on a bitmap.
LetterClassifier	This class extends Classifier and provides some functionality specific to letter recognition.
ID3Classifier	An implementation of a classifier based on ID3/decision trees.
NNClassifier	A neural network handwritten letter recognizer.
TrainClassifier	This program trains a classifier and saves it in a file to be read when used.
EvalClassifier	This program tests a classifier after loading it and the bitmaps.
UseClassifier	This program allows the user to draw on a bitmap.

bitmap

```
public Bitmap(int nRows, int nCols) {  
    map=new boolean[nRows][nCols];  
}
```

Class Summary	
Bitmap	A Bitmap holds a matrix of bits (true or false, on or off 0).
ClassifiedBitmap	A subclass of Bitmap and extends it by adding a classifier (a target class).
Classifier	A null model for classifiers acting on a bitmap.
LetterClassifier	This class extends Classifier and provides some functions specific to letter recognition.
ID3Classifier	
NNClassifier	
TrainClassifier	
EvalClassifier	
UseClassifier	

```
/**  
 * Classifies the bitmap  
 * @param map the bitmap to classify  
 * @return the probabilities of all the classes (should add up to 1).  
 */  
public double[] test(Bitmap map) {  
    return null;  
}
```

```
/**  
 * Classifies the bitmap  
 * @param map the bitmap to classify  
 * @return the probabilities of all the classes (should add up to 1).  
 */  
public double[] test(Bitmap map) {  
    double[] out=nn.feedforward(map.toDoubleArray());  
    return out;  
}  
  
/**  
 * Trains the neural network classifier on randomly picked samples from specified training data.  
 * @param maps the bitmaps which are used as training inputs including targets  
 * @param nPresentations the number of samples to present  
 * @param eta the learning rate  
 */  
public void train(ClassifiedBitmap[] maps, int nPresentations, double eta) {  
    for (int p=0; p<nPresentations; p++) {  
        int sample=rand.nextInt(maps.length);  
        nn.train(((Bitmap)maps[sample]).toDoubleArray(), targets[maps[sample].getTarget()], eta);  
    }  
}
```

bitmap

Class Summary	
Bitmap	A Bitmap holds a matrix of bits (true or false, on or off, 1 or 0).
ClassifiedBitmap	A subclass of Bitmap and extends it by adding a classification (a target class).
Classifier	A null model for classifiers acting on a bitmap.
LetterClassifier	This class extends Classifier and provides some functionality specific to letter recognition.
ID3Classifier	An implementation of a classifier based on ID3/decision trees.
NNClassifier	A neural network handwritten letter recognizer.
TrainClassifier	This program trains a classifier and saves it in a file to be read when used.
EvalClassifier	This program tests a classifier after loading it and the bitmaps.
UseClassifier	This program allows the user to draw on a bitmap.

```
public TrainClassifier(String[] args) {
    // create the classifier
    NNClassifier c=new NNClassifier(32, 32);
    // or - for ID3 - replace with the following line of code
    // ID3Classifier c=new ID3Classifier(32, 32);

    // load data
    try {
        ClassifiedBitmap[] bitmaps=LetterClassifier.loadLetters(args[1]);
        // train it using all available training data
        c.train(bitmaps,100000,0.01);
        // c.train(bitmaps);
    } catch (IOException ex) {
        System.err.println("Error loading data.txt: "+ex.getMessage());
    }
    try {
        Classifier.save(c, args[0]);
    } catch (Exception ex) {
        System.err.println("Failed to serialize and save file: "+ex.getMessage());
    }
}
```

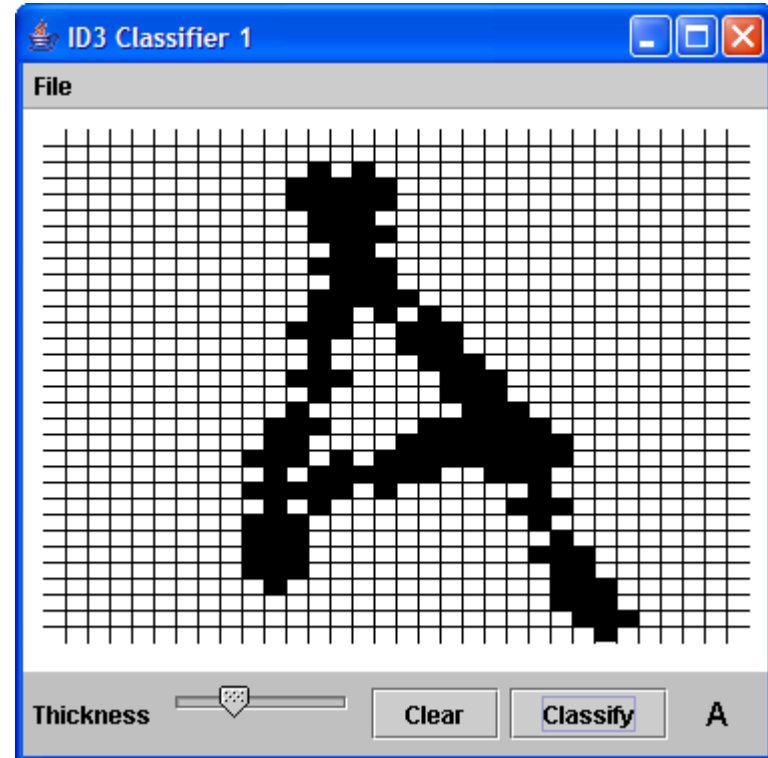
```
public static void run(Classifier c, ClassifiedBitmap[] bitmaps) {
    // test it using available data
    System.out.println("Evaluating classifier "+c.getName());
    System.out.println("Sample\tTarget\tActual\tCorrect");
    for (int i=0; i<bitmaps.length; i++) {
        int actual=c.index((Bitmap)bitmaps[i]);
        int target=bitmaps[i].getTarget();
        System.out.println(i+" \t"+c.getLabel(target)+" \t"+c.getLabel(actual)+" \t"+(target==actual?"YES":"NO"));
    }
}
```

```
/**
 * Classifies the bitmap
 * @param map the bitmap to classify
 * @return the probabilities of all the classes (should add up to 1).
 */
public double[] test(Bitmap map) {
    if (tree!=null) {
        double[] out=new double[getClassCount()];
        String actual=tree.getClassification(labels, map.toBooleanArray());
        for (int i=0; i<LetterClassifier.getClassCount(); i++)
            if (getLabel(i).compareToIgnoreCase(actual)==0)
                out[i]=1;
        return out;
    } else
        return null;
}

/**
 * Trains the ID3 classifier on provided samples.
 * @param maps the bitmaps which are used as training inputs
 */
public void train(ClassifiedBitmap[] maps) {
    features=new boolean[maps.length][];
    targetValues=new String[maps.length];
    for (int p=0; p<maps.length; p++) {
        features[p]=((Bitmap)maps[p]).toBooleanArray();
        targetValues[p]=getLabel(maps[p].getTarget());
    }
    id3=new machl.BinID3(labels, features, targetValues, classValues);
    tree=id3.induce();
}
```

bitmap

Class Summary	
Bitmap	A Bitmap holds a matrix of bits (true or false, on or off, 1 or 0).
ClassifiedBitmap	A subclass of Bitmap and extends it by adding a classification (a target class).
Classifier	A null model for classifiers acting on a bitmap.
LetterClassifier	This class extends Classifier and provides some functionality specific to letter recognition.
ID3Classifier	An implementation of a classifier based on ID3/decision trees.
NNClassifier	A neural network handwritten letter recognizer.
TrainClassifier	This program trains a classifier and saves it in a file to be read when used.
EvalClassifier	This program tests a classifier after loading it and the bitmaps.
UseClassifier	This program allows the user to draw on a bitmap.



machl

Class Summary	
BinID3	BinID3 contains methods for inducing a "binary" decision tree using Shannon's information theory.
BinTree	BinTree is a class for storing and generating binary trees (of nodes).
NN1	A basic implementation of a single-layered feedforward neural network and backpropagation learning.

Applications

- GenerateLetters
- TrainClassifier
- UseClassifier
- EvalClassifier

1: BASIC STUDY

You may complete one or both of the tasks 1A and 1B. Investigate and optimise the performance of the already implemented machine learning technique. Divide the original data set into at least two halves (a training and a test set). Describe the optimisations investigated. Describe and discuss the performance of the system on training and test data.

2: ADVANCED STUDY

You may complete one or both of the tasks 2A and 2B. Extend your chosen technique to produce a more advanced classifier. You are to modify the appropriate source code provided for the technique you have chosen. Evaluate your extension on the same problem. Describe both your implementation and its evaluation. Compare with the results in the BASIC STUDY.

3: COMPARE STUDY

To be eligible for marks in Part 3 you must have attempted all previous parts (i.e. 1A, 1B, 2A, and 2B). Compare the techniques, discussing whether one technique is easier to implement and whether one technique performs better in your implementation.

4: REFINE STUDY

Any number of the following tasks (4A-4D) may be completed. All are assigned the same marks. It is important to note that Parts 1 and 2 (i.e. [1A and 2A] OR [1B and 2B]) MUST have been attempted to be eligible for marks outlined in this section. You may attempt any part(s) of this section. Describe the method, your implementation, and the performance of your implementation. Compare with the results in the BASIC and ADVANCED STUDY.

5: DISCUSSION

Discuss how you could implement the BASIC and ADVANCED STUDY with another machine learning technique. Discuss other ways that you could improve the performance of the system. You should refer to published literature in this section, appropriately referencing this work, including a list of references.

6: COMPETITION

Enter your best classifier into the class competition to be tested on a new set of data.

1: BASIC STUDY

You may complete one or both of the tasks 1A and 1B. Investigate and optimise the performance of the already implemented machine learning technique. Divide the original data set into at least two halves (a training and a test set). Describe the optimisations investigated. Describe and discuss the performance of the system on training and test data.

1A: Neural Networks (3 marks)

Optimisations include the learning rate, the number of iterations of learning, and the size of the test/training sets.

1B: ID3 Decision Trees (3 marks)

Optimisations include the size of the test/training sets.

2: ADVANCED STUDY

You may complete one or both of the tasks 2A and 2B. Extend your chosen technique to produce a more advanced classifier. You are to modify the appropriate source code provided for the technique you have chosen. Evaluate your extension on the same problem. Describe both your implementation and its evaluation. Compare with the results in the BASIC STUDY.

2A: Multi-layer neural network (5 marks)

(See Russell and Norvig, p. 745-747.)

Add units to the single-layer neural network to give it two layers of weights.

Optimisations include the number of hidden units.

2B: ID3 with pruning (5 marks)

(See Russell and Norvig, p.662-663.)

Use any relevance-based heuristic to remove paths (“prune” decisions) in the decision tree.

Optimisations include the cut-off used for pruning.

3: COMPARE STUDY (1 mark)

To be eligible for marks in Part 3 you must have attempted all previous parts (i.e. 1A, 1B, 2A, and 2B). Compare the techniques, discussing whether one technique is easier to implement and whether one technique performs better in your implementation.

4: REFINE STUDY

Any number of the following tasks (4A-4D) may be completed. All are assigned the same marks. It is important to note that Parts 1 and 2 (i.e. [1A and 2A] OR [1B and 2B]) MUST have been attempted to be eligible for marks outlined in this section. You may attempt any part(s) of this section. Describe the method, your implementation, and the performance of your implementation. Compare with the results in the BASIC and ADVANCED STUDY.

4A: Create an Ensemble of Classifiers (4 marks)

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 18.4 on p. 664-668 for some background.

4B: Implement a pre-processing technique (4 marks)

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.

4C: Overtraining Prevention (4 marks)

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. See Russell and Norvig, p. 661-663 for some pointers.

4D: Additional Refinement (4 marks)

Implement some technical refinement that aims to improve the performance of the classifier and that has not been specified in 4A – 4C.

5: Discussion (3 marks)

Discuss how you could implement the BASIC and ADVANCED STUDY with another machine learning technique. Discuss other ways that you could improve the performance of the system. You should refer to published literature in this section, appropriately referencing this work, including a list of references.

6: COMPETITION (2 marks)

Enter your best classifier into the class competition to be tested on a new set of data.

- Will be run during the final lecture of the semester (Tuesday 27 October)
- NOTE: make sure that your classifier works as specified; classifiers that do not will not take part in the competition
- Will work as `bitmap.UseClassifier.java`
- Classifiers will be identified by the name you give it
- +1 mark for entering
- +1 mark for top 4 classifiers

2 Suggested Strategies

- Compare NN and ID3
- 1A (3)
- 1B (3)
- 2A (5)
- 2B (5)
- 3 (1)
- 5 (3)
- Refine NN or ID3
- 1A / 1B (3)
- 2A / 2B (5)
- 2 of 4A-D (8)
- 5 (3)
- 6 (1)

Submission

- Submit two files online (<http://submit.itee.uq.edu.au/>)
 - Source code (*.java into a .jar or .zip)
 - PDF of your report
- Group name should be the student number of the person submitting the assignment
- If you resubmit, please resubmit all relevant files

Submission

- Submit an assignment cover sheet signed by all members of the group to the COMP3702/7702 submission box on level 1 of GP south
 - Provided by the online submission system OR
 - <http://studenthelp.itee.uq.edu.au/assignments/>

Questions?