Assignment 3 – indicative solutions

1a: Order of importance: Hiddens → Iterations → State
The finding becomes clear when varying Hiddens while leaving states and iterations fixed, then by varying iterations while leaving hiddens and states fixed, and then by varying states while leaving hiddens and states fixed. States and hiddens should be varied within 3 and 25 neurons, while the iterations can vary from 1 to 10,000 (and more if a fast CPU is available). The analysis of the results reveals that the best cost benefit ratio is obtained when increasing the number of hidden layer neurons. This is followed by the number of iterations. Lastly, the system's performance is also slightly affected by the states though the computational cost does not justify the benefit in performance.
Maximum achievable performance is 100% on both training and test set.

1b: Changing the seed value can reduce the best performance achieved in 1a. This is because the parameters used in 1a (which achieved the best performance) were best for the given initial network condition. By changing the network condition, this affects the parameters which will produce the best performing system.

1c: The classification is better if the training set is a good representation (provides a good coverage) of the problem domain. If the training set is a suitably sampled subset of the domain then the performance of the training set can be at most equal to the performance of the training set. The test performance can be worse due to overfitting. The test performance can be better (than the training performance) if the test samples are not suitably covering the domain (i.e. if the training samples only cover a region which is easily learned).

1d: It is hard (but possible) to obtain 100% classification performance. This is due to some discriminating features being very similar, or are far away from the supervised root node. For example, the two classes 0 and 1 can only be discriminated by the label attached to one of the leaf nodes (the one representing the right hand), while the supervised node is the root node. It takes a long tome for the network to propagate back the error through the structure all the way to the furthest node (the leaf node).
Another reason could be (even so it is not the problem for this given learning problem) that perfect classification may only be obtain when we are near the minimum network error. However, the closer we get to the minimum error, the smaller is the gradient, and hence, the learning speed will slow down as we approach the minimum. However, Rprop addresses the issue, and hence, this later issue is not the reason for the difficulty of this learning problem.

1e: Training the system for several thousand iterations (several hours of training time on a fast computer) can help achive 100% on both, training and test set.
The reason to why the existing benchmark performance is less than 100% is due to the fact that computers were a lot slower 6 years ago. Training could not be done within a reasonable time to justify the minor improvement in performance. In fact, 6 years ago, the systems were only trained for at most 400 iterations. However, todays' computers are fast enough so that training can be done for many more iterations (as many as 10,000 iterations and more). This allows the system to converge to a lower error, producing a better classification rate.

2a: The dataset is severely unbalanced. The smallest class is class A which contains only 28 samples whereas class c contains 937 samples. Thus, the dataset is unbalanced by a factor 937/28=33.46

2b: An unbalanced dataset reflects a property of the learning domain: The occurrence of some data is more likely than the occurrence of data from some other class. The smaller the sample size of a

small class, the more likely it is that this data was generated by noise, and hence, the more likely it is that the system is to ignore such data. Moreover, the fewer samples we have for a given class the less certainty we have about the accuracy in the representation of the class. The reduced certainty causes the system to focus on classes for which there is greater certainty (classes which are represented by more training samples).
In contrast, a properly trained system can be tested by using an arbitrary number of test samples. The system makes no assumption about the certainty of a test sample. This is because the system does not know the class membership of a test sample, and hence, the test sample appears transparent to the system (it could belong to any class). The output of the system determines the class membership of a given sample.

2c: The learning rate can be increased for the smaller classes. For example, when updating the weights of the system for a pattern that belongs to class A then the learning rate should be 33.46 times larger than when updating the weights for a pattern that belongs to class c. the inverse is possible, too: Use a smaller learning rate for larger classes.
Another popular solution is to increase the samples of smaller classes by simply copying existing samples. For example, each training sample in class A would be copied 33 times so as to generate as many samples as there are in the much larger class c.

3a: Confidence of {a} → {b} = support(a,b)/support(a) = 0.2/0.25 = 0.8
The rule is interesting because the confidence of the rule exceeds the threshold value of 60%.

3b: Interest of {a,b} = support(a,b)/(support(a) * support(b)) = 0.2/(0.25 * 0.85) = 0.9412
The items are negatively correlated according to interest measure.

-END-