A Neural-Network system for Automatically Assessing Students

D.J. Mullier
D.J. Moore
[1]Faculty of Information and Engineering Systems
Leeds Metropolitan University
England
d.mullier@lmu.ac.uk
D.J.Hobbs
[2]University of Bradford
England

Abstract

This paper is concerned with an automated system for grading students into an ability level in response to their ability to complete tutorials. This is useful in that the student is more likely to improve their knowledge of a subject if they are presented with tutorial material at or just beyond their ability. However, dynamically responding to a student's changing knowledge about a subject usually requires the presence of a human teacher, an altogether expensive resource. The system discussed here can grade both a student and the questions in a tutorial with minimal input from the human teacher. In order to accomplish this a specialist neural network is employed. The design and operation of our system is discussed along with arguments as to why a neural network approach is suitable for this problem.

Keywords: automatic assessment, neural network, fuzzy logic

1 Introduction

The Tutorial Supervisor (TS) is an automatic system for grading a student into an ability level in response to the student's interaction with tutorial questions. Once the student has been graded then a question or tutorial can be selected which is inline with best pedagogic practice (Bergeron 1989). Our TS is an expansion of the TS designed by Bergeron et al (1989) and is in use in our prototype hypermedia system (Mullier et al 1999). Our TS improves on Bergeron's original specification by having the additional ability to adapt to both students and questions/tutorials as the system is in use, as opposed to requiring the system to be taken off-line and reprogrammed/trained. The TS's ability to respond in real-time to a student's changing ability and how a population of students perceive a particular tutorial or question is brought about by the use of a specialist neural network device that is able to learn and adapt without human intervention. A thorough description of our TS along with complete details of its design and testing can be found in Mullier (1999, chapter 8).

2 Operation of the Tutorial Supervisor

The TS is simple in operation. A set of questions to be given to the student is recorded in a database and is graded with a difficulty level by the author of the question. A student can then be given an appropriate question, depending upon the student's ability and the difficulty of the question. However, in order for a student to progress in their learning it is necessary to pitch questions so that they sufficiently tax the student without it being impossibly difficult (Bergeron 1989). Therefore it is necessary to track the student's change in ability as they progress through the learning material. Our system achieves this by recording the student's interactions with questions and mapping this onto an ability level. The system is robust to exceptions in the student's behaviour, since a student may generally perform well but make a mistake with one particular question. Similarly, the system is able to regrade questions in the question database. For example, a question may have been graded by the author as being relatively easy. However, it may transpire that a population of students actually find it difficult. This will be bourn out by most students who should have performed well with the question actually performing poorly. Such a situation negates the pedagogy stated above. Our system is able to statistically determine that a question has been misgraded and is able to remedy the situation. The remainder of this paper will discuss the issues relating to the design and implementation of our TS system.

3 Rationale for Using a Neural Network

A neural network is an Artificial Intelligence (AI) system that is able to learn rules in response to being presented with many examples. The neural network is said to learn the rules from the examples. In contrast a traditional rule-based system would have rules encoded within it that a designer has previously identified. The advantage of neural network systems is that it is not always possible for a human designer to express and encode rules in a reasonable time-frame or even express then at all. A further disadvantage of rule-based systems is that if the rules change for some reason then it is necessary for the designer to reincorporate the new rules within the rule-base (Hagan et al 1996).

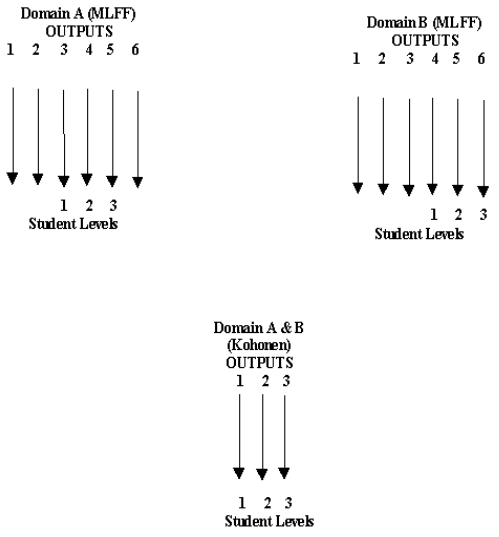

Figure 1 Kohonen versus MLFF neural networks

A further reason for choosing a neural network for the TS in preference to a rule-based system is that, unlike a rule-based system, a neural network can be domain independent. It is unlikely that, for example, a high level student produces results in the same range for every type of domain. Thus the rule "IF SCORE >70 THEN LEVEL 10" is only likely to apply to the domain that it was initially defined for. This is the reason why Bergeron et al (1989) use a neural network for their Tutorial Supervisor. Their neural network holds the rules that it has learnt from its training data (the first domain). It is then able to change its rules in response to new data (new domains), by retraining off-line. In this manner the neural network can adapt to misconceptions or inaccuracies in the original rules and adapt to new situations. This would be a difficult and time-consuming process for a symbolic rule-based system, since it would require the re-engineering of the rule-base by a designer, in that the new rules would have to be identified and then encoded. In essence, the neural network is doing the job of the human rule designer. Designing such rules is not necessarily a simple matter, since it requires the human designer to examine many student interactions with various questions and tasks so that a valid grading of each question can be made (e.g. this question was answered well by novice students, it is therefore easy and can be presented to other novice students). The situation is further complicated by the possibility that different populations of students (students from different classes or tutorial groups) may have different previous knowledge of the domain and therefore the initial question gradings may not apply to them. It would therefore be helpful if an AI system could be employed to accomplish the task of dynamic question grading and therefore remove some burden from the author. However, Bergeron et al's (1989) TS is unable to readapt to different domains without being manually provided with new training data and then retrained off-line. The remainder of this paper describes the neural networks used for our Tutorial Supervisor, which improves upon Bergeron et al's (1989) design by allowing the automatic on-line adaptation to different domains.

4 Tutorial Supervisor Architecture

The Kohonen self-organising map is a specialist type of neural network with the ability to learn without human intervention (Hagan et al 1996). This is useful in our system since the distinction between a novice student and an expert student, in terms of marks at tutorial nodes, may be small, or may vary significantly from domain to domain. For example, the majority of students may achieve marks between 50% and 60%, with a few results between 60% and 75% percent and a few between 40% and 50%. There are therefore two large ranges of numbers that occur infrequently (0-40 and 75-100). If a standard type of neural network were to be used to model the above problem then these mark ranges must be identified beforehand or the neural network's outputs would have to be designed to produce student ability levels between 0 and 100, in order to accommodate a broad range of scores. Identifying scores beforehand is not likely to be practical since it would require the collection of a large amount of data (with no initial benefit for the student). Designing such a generic neural network also introduces the following difficulties. The standard neural network must have enough ability levels (outputs) to clearly demonstrate the distinction between students in these highly clustered areas, necessitating an increase in outputs for all areas to cover for all possibilities, even those that are unlikely. The increase in outputs renders the neural network more complex, resulting in a network that is more difficult to train. Further, and most crucially, once the trained standard neural network is used for different domains, then there is no direct correspondence between an ability level for one domain and an ability level for another. This is because an output of the standard neural network does not correspond directly to a student ability level, since the student ability level may vary between domains. The upshot of this is that the standard neural network would require complete retraining for different domains.

A Kohonen network can solve the above problem by continually adapting to input stimuli whilst it is being used by students. This is because of the way a Kohonen network operates. A Kohonen network is given a number of outputs by the network designer, representing the number of categories that the network designer wishes the network to identify (the number of required student levels in our case). It is left to the network itself to sort the input data into this number of categories, since it is not implied by the training data itself. In the case of a standard neural network it would be required to include an example solution with the training data. If there are ten or more distinct patterns in the data then a correctly trained Kohonen will learn by itself to distinguish them (Kohonen 1989, Masters 1993, Gurney 1997). Therefore, it is irrelevant to the network if the actual values of the input data change; it will still attempt to separate them into the number of categories represented by the number of outputs that it has. The outputs of a Kohonen neural network therefore behave as fuzzy sets, whose boundaries may change over time. The Kohonen neural network is therefore simpler than the standard neural network for the TS in this case, since the number of outputs can be kept small since is not necessary to design for all unlikely possibilities (the network will adapt to them if they occur). This is shown in figure 1. Two standard neural networks, in our test case Multi-Layered Feed-Forward (MLFF) neural networks, are shown in operation with different domains where the scores of tutorials are different. In order for the neural network to grade both domains it needs enough outputs to cope with the two cases, even though only a subset of the outputs are used for an individual domain. By contrast the Kohonen neural network needs only the number of student levels required, as it will adapt to the differing gradings between domains.

Training Data

A neural network has a number of inputs, which in our case represent the student's responses to questions and a number of outputs, each one representing a unique ability level. The inputs to the Kohonen neural network must incorporate history data in order to make a more informed evaluation of the student and therefore avoid a restriction of Bergeron at al's (1989) neural network, namely reacting to a one off error (or success) from a student. History data can be used to prevent the TS from making snap judgements on the student. For example, if the student is generally performing well, but gets one question wrong, then if no history data is taken into account the TS is forced to make a decision based only upon the most recent presentation and the student is likely to drop a level. The student ability itself represents a degree of history data, in that if a student is regarded to be an expert student, then they must have performed well in the past. However, the direct incorporation of history data prevents a continual changing of levels based upon one interaction only. The incorporation of history data can be achieved by presenting a number of previous interactions with tutorial nodes to the neural network. Each time a new interaction is presented the previous interactions are shifted along the inputs to accommodate the new input and the oldest interaction is lost. Training data supplied to the neural network are figures that represent a percentage value of a student's interaction with a tutorial. For example, if the student achieved a 50% success level with a tutorial question, then it is this figure that is passed to the TS.

Re-grading Questions Using Fuzzy Logic

Each question level is generally presented to a student of the same level, or just below, a pedagogy used with success by Bergeron et al (1989), in that a level *x* student should be able, overall, to answer a level *x* question. A question may however, be graded incorrectly by the domain author. This can be determined by the system after a number of interactions with different students (a population of students who should be, generally, getting a question right are getting it wrong or vice versa). It is not suitable to immediately re-grade a question with respect to an interaction with one student, however. As has been discussed earlier, a student is a complex entity and it is difficult to formulate rules describing them accurately. In order to resolve this problem each question level is modelled as a fuzzy set. This allows a question's level to be adjusted slightly, within the level, without necessarily affecting the overall level (as presented to the student), thus there is a buffering effect and the question does not rapidly leap back and forth between levels. The use of fuzzy sets also provides a mechanism for allowing a question to belong to more than one question level set, providing a smoother transition between levels. This differs from Bergeron et al's (1989) approach in that they collect data from the students and then periodically use it to update the training of the neural network. There is therefore a delay, which ensures that the question levels do not suddenly change, which could potentially result in the question level continually changing and thus be distracting to the students. However, the drawback is that this is a manual process that requires the direct intervention of the system designer. The process for regrading questions described below is achieved automatically, whilst still maintaining the delay between the question being presented to a student and changing its level.

A question is re-graded by a population of students' interactions with the question being determined as incorrect by the TS, for the reasons described above. Such erroneous interactions cause the question ability to move within the fuzzy set until it crosses into a different fuzzy set.

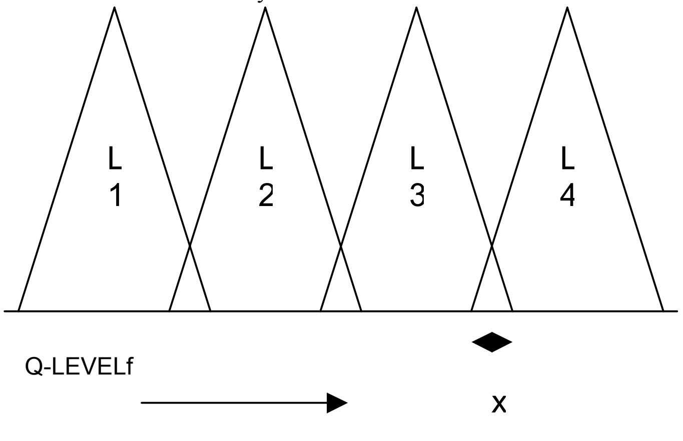

Figure 2 Regrading questions with fuzzy sets

A question's ability level is therefore only changed after a number of erroneous interactions with students, the actual number being dependent upon the size of the fuzzy set, the fuzzy set is therefore acting as a buffer. A simple fuzzy processor accomplishes question re-grading. The fuzzy processor compares the level of the current question and the student level output of the TS neural network. The buffer can be implemented using three fuzzy rules:

1 IF S_LEVEL > Q_LEVEL THEN
Q_LEVELf = Q_LEVELf + 1

2 IF S_LEVEL < Q_LEVEL THEN
Q_LEVELf = Q_LEVELf – 1

3 IF S_LEVEL = Q_LEVEL THEN
Q_LEVELf = Q_LEVELf (remain unchanged)

where
S_LEVEL is the level assigned to a student
Q_LEVEL is the level of the question, used to decide whether it is suitable for the student.
Q_LEVELf is the fuzzy membership number of the question.

Using the figure 2, a question may belong to one or two of four levels. If the question has a value of Q_LEVELf corresponding to x, then the question is regarded as both level three and level four. If however, as a result of interactions with several students, rule one is repeatedly fired, then the value of Q_LEVELf will increase and the question will become graded as level four only. Conversely, if rule two is repeatedly fired then the value of Q_LEVELf will decrease and the question will be graded as level3 only. Such changes may result in further increases or decreases in the value of Q_LEVELf, which may become any of the levels available. The utilisation of this fuzzy system ensures that the level of a question is not changed (in terms of presenting to students) in response to individual interactions with students. The fuzzy logic is able to distinguish overall trends and is therefore robust in the presence of exception data.

The changing of levels is dependent upon the size of the fuzzy sets, since the size of the fuzzy set directly affects how many interactions are required before a question migrates from one level to another. These start and end points also define whether a question can belong to more than one level or not. It is intuitively sensible to provide a small overlap of neighbouring fuzzy sets. This provides a simple buffer that will help the question to find its true grade. For example, if fuzzy sets were to be defined separately (or as a discrete set) then a question may be presented as a difficult level but become graded as an easier level after several interactions. Once this change has occurred then the question is no longer presented to the original class of students that graded it. This may result in the question becoming stuck at a particular level. If instead the question is presented to both the original class of student and the new class of student, then a smoother progression from one level to another may result and the extra information enables the question level to settle more easily. If the fuzzy sets are defined so that a question may belong to more than two levels, then the question level may not settle at all, as one level of student may push the value of Q_LEVEL one way and another level may push it in the opposite direction.

5 Experimental Trials

Our TS was designed using a Neural Network package, NeuroShell2 by Ward Systems. Several configurations of neural network were built and tested with both simulated data and real student interactions. The most successful architecture is discussed below. A full description of the experimental trials can be found in Mullier (1999, chapter 8).

The Kohonen neural network proved to be a successful neural network architecture for the problem of grading students into ability levels. Most permutations of parameters produced neural networks that converged upon a solution. A fast and reliable neural network could be produced with between five and twenty inputs or outputs. It is possible to increase this number, but this is unlikely to be required, since it is not desirable to use information from too far in the past, since the network does not have any knowledge of time. It has been experimentally determined that the number of outputs should not rise above twenty. If more levels are required then the outputs may be combined to form fuzzy sets.

Parameters for a generic Kohonen Tutorial Supervisor, i.e. one that will converge on a solution and provide a high degree of student grading for a variety of domains are suggested as the following:

- 5 Inputs – this has been found to provide the network with enough information with which to evaluate the student. It is suggested that an input filter, as described above is employed if more than this number of inputs is required.

- 10 Outputs – corresponding to ten student levels, if more levels are required then it is advised that thirty levels is the upper limit, beyond this the neural network becomes less likely to activate all of its outputs.

- Any form of data extraction may be used to form the training set, however a large number of examples are required to produce a fully trained network. One thousand student interactions provide enough examples. Note therefore, that for a network to be trained with real student data is probably impractical. However, since the Kohonen network is fully adaptable, it is suggested that a network be trained upon generated data and then allowed to adapt to real students. The generated data could be designed to represent realistic but uncomplicated situations. For example, training the network to model simple rules such as “If result in the range 40-50 THEN set student level to 5”. The neural network is then able to adapt to any misconceptions present in these rules. Once a neural network has been trained it may be saved and replicated.

6 Discussion and Conclusion

The Tutorial Supervisor is intended to be an automatic system for gauging a student’s abilities with tutorials. This imposes a restriction upon the kind of material that can be offered in a tutorial, since it must be suitable for automatic assessment. Automatic assessment effectively rules out assessments that cannot be graded in a relatively simple fashion. For example, it would not be possible, with current technology, to have an essay automatically assessed, since this would require a complex understanding of the essay on the part of the assessor. Automatic assessment is limited to tasks that can be broken down into elements that can then be individually marked. A tutorial may, for example, take the form of ten questions, each of which could be answered by the student and graded as right or wrong, or graded as containing relevant keywords. However, assessment is not the role of the TS, which is presented with completed assessments (results). Assessment is therefore limited to multiple-choice questions or identifying that the student has visited certain nodes (or a combination of both).

A key issue of concern regarding the TS is the number of student levels that the TS is to recognise and output. Each student level should have tutorial material generated for it; since it is important to target tutorial tasks at the student's ability, this is seen as being of more educational benefit than offering the same tutorials to all students and then assigning a student level based upon the grade that the student achieves (Bergeron 1989), although there is no technical reason why the latter could not be done. The number of student levels therefore may change between domains, since some domains may have a richer set of assessment questions than others (for a number of possible reasons). A possible conflict therefore could arise between the number of student levels that has been designed into the TS by the system designer and the number of student levels that are required by the current domain author. A possible solution to this problem is for the system designer to provide a TS that is capable of outputting a large number of student levels and then each domain author can allow it to adapt to their domains and ignore the inactive outputs from the TS that will naturally arise if there is not sufficient input student levels. The benefit of this approach is that one TS configuration could be used for many domains without the need for reconfiguration. However, the drawback is that some outputs of the TS will always remain inactive, although the experiments carried out as part of the research demonstrated that active outputs tend to cluster together and so are easily identifiable. A problem related to the number of outputs is the number of inputs.

The amount of history data presented to the TS directly affects the grading of the student, in that the more history data presented to the neural network, the greater the effect of previous results with tutorials. This is a similar situation to that of the number of student levels, in that it is possible to design a TS with a large number of inputs and then use only the required amount. However, it is not a simple matter to determine how much history data to present to the neural network in order to aid the student the most. This issue is difficult to reconcile without extensive trials with real students and even if this were done it would still be unlikely that any real conclusions could be drawn since proving the effectiveness of educational systems is notoriously difficult in the educational field (Dillon and Gabbard 1998). The purpose of the TS here is to explore the technical issues relating to the feasibility of providing an automatic student grading system. Whether this facility is useful is open to educational debate. However it is likely to be the case that it will be useful should the correct set-up of the TS be achieved during trials with real students, since Bergeron et al (1989) found their TS to be useful.

Further issues arise concerning the adaptability of the neural network used for the TS. The neural network architecture used by Bergeron et al (1989) requires off-line training and is therefore under the control of the system designer. The drawback with this approach is that it requires the manual intervention of a person who can interpret the student interaction data with tutorials and determine whether it should be represented to the neural network. The advantage of the Kohonen neural network architecture is that it is able to train continually without any intervention from a human. However, there are situations where this adaptation is undesirable, most notably when different skill levels of students use the same domain at different times. For example, if a class of first year students use the system followed by a class of final year students. However, this is not a problem if the questions and tutorials have been adequately assigned a difficulty level, since the first year students will only be offered easier tutorials and so can be graded only as lower level students (although they can still progress if they continue to achieve success with the tutorial). Problems can arise only if both the student abilities and the question difficulties are unknown beforehand. This is because the TS acts as a bi-directional mapping device, in that if either the student abilities or the question difficulties are known beforehand then the TS can produce the unknown parameter. It is not, however, able to produce values when nothing is known beforehand. The TS's ability to re-grade questions automatically is an exploitation of this bi-directional mapping facility, in that the student ability can be changed in response to improving results and the question difficulty can be changed if a significant proportion of students who should get the question right in fact get it wrong.

Research into the TS has demonstrated that a fully adaptable system for automatically grading students is possible and practical. The approach of using an automatic tutorial supervisor has been practically justified by Bergeron et al (1989). However, their system requires manual periodic retraining which renders it unsuitable for a generic tutorial system, or a tutorial system that can be used without the need for reprogramming or otherwise rearranging the program code of the system.

Further research is concerned with incorporating the TS within a hypermedia tutoring system (Mullier et al 1999, Mullier 1999) so that the students' interactions with the TS can be studied. It is anticipated that such a study will prove useful for determining how the TS reacts to different domain, where the rules that describe ability are different, with a view to reengineering the TS so that it is able to learn such a vast set of rules without conflict. A "superTS" such as this would be useful in a more generic tutoring environment such as may become more prevalent on the WWW.

6 References

Bergeron B, A Morse, R Greenes, 1989, "A Generic Neural Network Based Tutorial Supervisor for C.A.I". In 14th Annual Symposium on Computer Applications in Medical Care. IEEE Publishing, pp 435-439

Gurney K 1997, "An Introduction to Neural Networks", UCL Press; ISBN: 1857286731

Hagan M T, H Bemurth, M Beale, 1996, "Neural Network Design", Pws Publishing Co; ISBN: 0534943322

Kohonen T, 1989, "Self-Organisation and Associative Memory". Springer-Verlag; ISBN: 3540620176.

Mullier D J, D J Hobbs, D J Moore, 1999," A Hybrid Semantic/Connectionist Approach to Adaptivity in Educational Hypermedia Systems". In Proceedings of ED-MEDIA'99, Seattle, WA. 1999

Mullier D J, 1999, "The Application of Neural Network and Fuzzy-Logic Techniques to Educational Hypermedia". PhD thesis. Available from www.lmu.ac.uk/ies/comp/staff/dmullier