Snap-Drift Neural Computing for Intelligent Diagnostic Feedback

Samson Habte

London Metropolitan University

A thesis submitted in partial fulfilment of the requirements of London Metropolitan University for the degree of

Doctor of Philosophy

June 2017

Acknowledgements

First and foremost I offer my sincerest gratitude to my supervisor, Professor Dominic Palmer-Brown, who has supported me through out my thesis with his patience and knowledge while allowing me the room to work in my own way. I could not have imagined doing my Ph.D study on modal learning neural networks if I were not given the opportunity. I am grateful for putting his trust in me to conduct my research in this area.

I would like also to thank the rest of my thesis supervisors: Dr. Fang Fang Cai, Dr. Miao Kang, and Professor Christina Jayne, for their insightful comments and encouragement, but also for the precious time and effort they put to widen my research from various perspectives.

I thank my fellow research students for the stimulating discussions and for all the fun we have had during my stay at London Metropolitan University. I thank also the academic staff of the school of Computing at London Metropolitan University, in particular I am grateful to Dr. King Fisher and Dr. Stan Zakrzewski for providing assistance in conducting my research experiment trials.

Last but not the least, I would like to thank my family: my parents and to my brothers and sisters, my cousin Gebrehiwet Mahari, and my friends for supporting me morally and materially throughout my Ph.D study.

Abstract

Information and communication technologies have been playing a crucial role in improving the efficiency and effectiveness of learning and teaching in higher education. Two decades ago, research studies were focused on how to use artificial intelligence techniques to imitate teachers or tutors in delivering learning sessions. Machine learning techniques have been applied in several research studies to construct a student model in the context of intelligent tutoring systems. However, the usage of intelligent tutoring systems has been very limited in higher education as most educational institutions are in favour of using virtual learning environments (VLEs). VLEs are computer-based systems that support all aspects of teaching and learning from provision of course materials to managing coursework. In this research study, the emphasis is on the assessment aspect of VLEs.

A literature review revealed that existing computer-based formative assessments have never utilised unsupervised machine learning to improve their feedback mechanisms. Machine learning techniques have been applied to construct student models, which is represented as categories of knowledge levels such as beginning, intermediate and advanced. The student model does not specify what concepts are understood, the gap of understanding and misconceptions.

Previously, a snap-drift modal learning neural network has been applied to improve the feedback mechanisms of computer-based formative assessments. This

study investigated the application of snap-drift modal learning neural network for analysing student responses to a set of multiple choice questions to identify student groups. This research study builds on this previous study and its aim is to improve the effectiveness of the application of snap-drift modal learning neural network in modelling student responses to a set of multiple choice questions and to extend its application in modelling student responses gathered from object-oriented programming exercises.

A novel method was proposed and evaluated using trials that improves the effectiveness of snap-drift modal learning neural network in identifying useful student group profiles, representing them to facilitate generation of diagnostic feedback and assigning an appropriate diagnostic feedback automatically based on a given student response. Based on the insight gained into the use of this novel method, we extend it to identify useful student group profiles that represent different programming abilities for writing an object-oriented class. The purpose of identifying student group profiles is to facilitate construction of diagnostic feedback that improves the development of basic object-oriented programming abilities.

Overall, the main objectives of this research project were addressed successfully. New insights are gained into the application of unsupervised learning in general and snap-drift modal learning in particular. The proposed methods are capable of improving the feedback mechanisms of existing computer-based formative assessment tools. The improved computer-based formative assessments could have a huge impact on students in improving conceptual understanding of topics and development of basic object-oriented programming abilities.

Contents

List of Figures

Glossary

Curse of dimension : Refers to the fact that algorithms are simply harder to design in high dimensions.

Euclidean distance : A straight line distance between two points and is equal to the root of square difference between coordinates of the two points.

Hamming distance : A number of bits which differ between two binary strings.

Iris data set : A popular data set in machine learning which contains 3 classes of 50 instances each, where each class refers to a type of iris plant.

Intermediate states : States in a state transition diagram which are between the start and the end states.

Learning rate : Defines by how much weight vectors are changed during machine learning.

Machine Learning : The science of getting computers to act without being explicitly programmed.

Regular expression : A special text string for describing a search pattern.

Scalar index of performance : Scalar feedback signal that indicates how well an agent is doing in the context of reinforcement learning.

State transition diagram : Technique to define a machine that has a number of states, which can change from one state to another state when it receives an

event from the outside world.

Synaptic weight : Strength or amplitude of a connection between two neurons or nodes.

Chapter 1

Introduction

Information and communication technologies have been playing a crucial role in improving the efficiency and effectiveness of learning and teaching in higher education. Two decades ago, research studies were focused on how to use artificial intelligence techniques to imitate teachers or tutors in delivering learning sessions. Machine learning techniques have been applied in several research studies to construct a student model in the context of intelligent tutoring systems [30; 44; 59; 68]. However, the usage of intelligent tutoring systems have been very limited in higher educations as most educational institutions are in favour of using virtual learning environments (VLEs), which provide a set of software tools to support learning and teaching. Examples of VLEs are Moodle, WebCT and Blackboard.

According to the survey conducted by Universities and Colleges Information Associations (Ucisa), 34% of all higher education institutions in the UK used Blackboard in 2001 and the usage increased to 60% by 2012 [66]. The survey also indicated that "Blackboard is the most used enterprise or institutional VLE, but Moodle has increased in usage as an enterprise solution and remains the most

commonly used VLE platform when departmental/school implementations are also considered". The main focus of VLEs is to support and facilitate teaching and learning, whereas intelligent tutoring systems aim to emulate teachers or tutors in delivering learning lessons. Previous intelligent tutoring systems were focused only on acquiring well defined procedural skills [57]. They did not address development of conceptual understanding.

Several computer-based assessments, which can be deployed as part of VLEs or independently, have been proposed that support different assessment types such as multiple choice questions [31; 55; 67], short-free-text response[35; 61; 62] and problem solving exercises [11; 26; 48; 64]. A literature review on the feedback mechanisms of existing computer-based formative assessments revealed that they only provide item-based feedback mechanism, which is a feedback tied to individual question or feature. Even though machine learning techniques were applied to construct student models as part of intelligent tutoring systems, the student model has never been used to facilitate construction of diagnostic feedback that improve student learning experiences in the context of formative assessments. Since the student model represents knowledge levels as categories such as beginner, intermediate and advanced, it is not suitable for facilitating generation of diagnostic feedback.

A snap-drift modal learning neural network have been applied to improve the feedback mechanism of computer-based formative assessments [4; 53]. These previous research studies investigated the application of snap-drift modal learning neural network for analysing student responses to multiple choice questions. The purpose of the analysis was to identify student groups that facilitate generation of diagnostic feedback. This research builds on these previous research studies in

order to improve the effectiveness of the application of snap-drift modal learning neural networks for diagnostic feedback based on multiple choice based student responses and extend its application to programming exercise-based student responses.

1.1 Aims and Objectives

The first aim of this research project is to improve the effectiveness of the previous application of snap-drift modal learning neural networks for diagnostic feedback in the context of VLEs. To achieve this aim, the following objectives are defined:

1. To analyse the learning behaviour of the snap-drift modal learning neural networks.

2. To propose a method for representing outputs of the trained snap-drift modal learning neural networks that facilitate generation of diagnostic feedback.

3. To propose criteria for assessing the effectiveness of student groups representing the different categories of understanding distinctly and facilitating generation of diagnostic feedback.

4. To evaluate the effectiveness of snap-drift modal learning neural networks in identifying useful student groups that can map to the different understanding levels of a particular topic.

5. To evaluate the impact of diagnostic feedback generated based on profiled student groups on improving learning performance of students.

There are many forms of assessment where student responses can be captured. The most popular assessment type is multiple choice questions(MCQs), which can be designed with a diagnostic end in mind in order to find out whether specific areas of a given subject are adequately known, or in order to detect misconceptions [17]. However, MCQs have limitations in assessing student's ability to apply techniques and solve problems. Problem solving exercise is another form of assessment where student responses can be captured, which can assess students' ability to apply techniques or to solve problems.

The second aim of this research project is to extend the application of snap-drift modal learning neural network for modelling student responses to object-oriented programming exercises. Based on the insight gained from the result of the first aim, the following objectives are defined:

1. To explore the different approaches for teaching and learning object-oriented programming languages.

2. To propose a method for representing a student response to object-oriented programming exercises in order to make it suitable to analyse using snap-drift modal learning neural networks.

3. To evaluate the effectiveness of snap-drift modal learning neural networks in identifying useful student groups that can map to the different object-oriented programming abilities of a particular programming exercise.

4. To evaluate the impact of diagnostic feedback generated based on profiled student groups on improving acquisition of basic object-oriented programming abilities.

1.2 Scope and Significance of Study

VLEs are computer-based systems that support all aspects of teaching and learning from provision of course materials to managing coursework. Most VLEs provide four software tools, which are content management and sharing, assessment management, collaboration and communication tools. The emphasis in this research is on the formative assessment aspect of VLEs, whose purpose is to provide feedback to students in order to enhance their learning.

This research contributes to knowledge on advancing the application of unsupervised machine learning techniques in general and snap-drift modal learning neural computing in particular for improving the feedback mechanism of formative assessments. Neural computing techniques have been applied successfully in various domains of application. In this project a new application domain, which is modelling observed student responses to multiple choice questions and Java programming exercises for facilitating construction of diagnostic feedback, is investigated.

Furthermore, the outcome of this research will have a significant impact on improving the learning experience of students by facilitating conceptual understanding of topics and development of object-oriented programming abilities.

1.3 Research Approach and Methods

There are two approaches to design and develop intelligent systems. The first approach is rule-based expert system, which is a knowledge based system that contains both declarative and procedural knowledge to emulate the reasoning

processes of human experts in a particular domain [45]. The second approach is machine learning such as neural networks, Bayesian networks and decision trees.

In this research, the focus is on neural networks, which are computational models that use ideas inspired from how human brains work. The fundamental difference between neural network systems and expert systems is the way knowledge is acquired. The source of knowledge in the case of expert systems is a domain expert, an individual selected for expertise in a given field. Knowledge is acquired in neural networks from previous examples, which are represented as training data. One of the drawbacks of rule-based expert systems, when they are applied to model student behaviours, is that they require eliciting the relevant domain and pedagogical knowledge from experts, a process that is often hard and time consuming [5]. The knowledge acquired from domain experts can typically recognize and interpret only expected student behaviors, and are unable to handle unanticipated ones.

Neural networks not only learn from experience, they can also generalise that means a trained neural network can produce reasonable outputs from inputs not encountered during its learning process [29]. Neural networks are able to derive meaning from complicated and/or imprecise data and to extract patterns that are too complex to be noticed by other computational techniques [25]. These characteristics make neural networks a powerful method to model human behavior and a useful technique to create user models for hypermedia applications [25].

Data collection is very important for training neural networks. Ideally, data should be gathered from every member of the population you are interested in. In this research, the population is the set of all students who are registered for a module whose topic is to be assessed. However, only students who register for

the module offered by London Metropolitan University are considered due to time and resource limitations. Both paper-based and web-based assessment sessions are used to collect student responses. To ensure the reliability of the collected data, the assessment sessions are carried out in an environment where students are not allowed to copy each other and get help from others.

1.4 Structure of the Thesis

The remainder of the thesis is organised as follows. Chapter Two provides a literature review on student modelling, computer-based formative assessments and neural computing. Chapter Three presents analysis of the previous application of snap-drift modal learning neural networks for diagnostic feedback in the context of virtual learning environments and how it is improved and evaluated. Chapter Four explores how snap-drift modal learning neural networks could be applied for modelling student responses to object-oriented programming exercises. Chapter Five presents analysis and evaluation of the proposed methods for identifying student group profiles that facilitates generation of diagnostic feedback to improve development of basic object-oriented programming abilities. Finally, conclusion and recommendations for future research are described in Chapter Six.

Chapter 2

Literature review

This chapter presents first an overview of neural computing. It introduces neural computing in general and focuses on unsupervised learning techniques. Secondly, it presents different student modeling techniques in the context of Intelligent Tutoring System (ITS). Thirdly, a brief review on computer-based formative assessment is described. The review focuses on how feedback process of formative assessments is supported or automated using ICT for three different assessment tasks: multiple choice questions (MCQs), short free text responses and problem solving exercises. For formative computer-based assessments, marking or scoring is less important, as their purpose is to improve student learning via feedback. That is why the focus is on feedback process not on automated marking.

2.1 Overview of Neural Computing

Neural computing is a computational model that uses ideas inspired from how human brains work. Neural network is a network of neurons which is found

in brains. Artificial neurons are crude approximations of the neurons found in brains. They can be implemented as a hardware or a software. Artificial neural networks (ANNs) are networks of artificial neurons. From a practical point of view, ANNs are just a parallel computational system consisting of many simple processing elements connected together in a specific way in order to perform a particular task [15]. In the functional level, ANNs resemble the brain in two aspects [29]. Knowledge is acquired by the network through a learning process and inter-neuron connection strengths known as synaptic weights are used to store the knowledge.

Neurons can be modelled by a set of three basic elements: [29]

1. A set of synapses or connecting links
2. An adder for summing the input signals, weighted by the respective synaptic strengths of the neuron.
3. An activation function for limiting the amplitude of the output of a neuron.

Mathematically, the neuron model is described by a pair of equations:

$$u_k = \sum_{j=1}^{m} w_{kj} x_j y_k = \varphi(u_k + b_k) \tag{2.1}$$

Where x_j are the input signals; w_{kj} are respective synaptic weights of neuron k; u_k is the linear combiner output due to the input signals ; b_k is the bias; $\varphi(.)$ is the activation function; and y_k is the output signal of the neuron.

There are three fundamentally different types of neural network architectures [29]. The first one is single-layer feed-forward where an input layer of source nodes project directly into an output layer of neurons. The second type is multilayer

feed-forward network, which consists of one or more hidden layers between an input layer of source nodes and an output layer of neurons. The third type is recurrent networks that contains a feed-forward single layered or multi-layered networks with at least one feedback loop.

Unlike expert systems which incorporate a knowledge base, neural networks do not have such a collection of information. They need to be trained for a given problem or situation so that the weights will then contain the required knowledge. Neural network learn by adapting the strengths/weights of the connections between neurons so that the final output activations are correct. The training data used by neural network to acquire knowledge can be labelled or unlabelled [29]. In labelled training examples, each example representing an input signal is paired with a corresponding desired response. On the other hand, unlabelled examples consist of different realizations of the input signal all by itself. Labelled examples may be expensive to collect, as they require the availability of a teacher to provide a desired response for each labelled example. In contrast, unlabelled examples are usually abundant as there is no need for supervision.

Neural networks can learn from training experiences using three different approaches: supervised, unsupervised and reinforcement learning [29]. Supervised learning relies on the availability of a training sample of labelled examples, with each example consisting of an input signal and the corresponding desired response. On the other hand, unsupervised learning relies solely on unlabelled examples, consisting of a set of input signals. In reinforcement learning input-output mapping is performed through the continued interaction of a learning system with its environment so as to minimise a scalar index of performance. Recently, a hybrid learning known as semi-supervised is emerging, which employs a training sample

that consists of labelled and unlabelled examples.

Single layer feed-forward neural networks can not deal with non-linearly separable problems. These type of problems can only be dealt by adding at least one hidden layer between the input and output layer. Multi-layered neural networks are trained using a weight update rule known as back-propagation [29]. It is a supervised learning technique that includes two passes, which are forward and backward. The forward pass propagates input data through the network to provide outputs at the output layer. In the backward pass, error values are propagated in reverse direction through the network to determine how the weights are to be changed during training.

The back-propagation algorithm can be summarised as follows:

- Repeat until the network converges.
  - For each input data.
    1. Perform a forward pass to find the actual output.
    2. Obtain an error values by comparing the actual and target output.
    3. Perform a backward pass of the error values.
    4. Use the backward pass to determine weight changes.
    5. Update weights.

Supervised learned ANNs can be applied to solve regression and classification problems. It has been applied successfully in several domains of applications such as financial modelling, time series prediction, computer games, control systems and pattern recognition [15]. For example, they can be applied in the area of financial modelling to predict stocks, shares and currency exchange rates, and in the pattern recognition area for speech and hand-writing recognitions.

There are several practical challenges of applying supervised learning ANNs to solve regression and classification problems [15]. The first challenge is choosing the optimal number of hidden layers and number of units within each hidden layer. The best number of hidden units depends on many factors such as the number of training patterns, the number of input and output units, the amount of noise in the training data, the complexity of the function or classification to be learned, the type of hidden unit activation function and the training algorithm.

Too few hidden units will generally leave high training and generalisation errors due to under-fitting where as too many hidden units result in low training errors, but will make the training very slow and will result in poor generalisation due to over-fitting. The sensible strategy for choosing an optimal number of hidden units is to try a range of numbers of hidden units and choose the network that performs best.

In principle, raw input data can be used directly to train neural networks, however, it will produce poor results if the raw input data is not transformed into some new representation [15]. Choosing an appropriate pre-processing technique is another challenge for many practical applications. The simplest pre-processing technique is a form of linear transformation of the input data and more complex pre-processing involves reduction of dimensionality of input data.

Finally, once the topology and methods of pre-processing are chosen, the challenge is choosing the right learning rate. It is not easy to choose an appropriate learning rate because of two opposing facts. If a learning rate is too small, the network will take too long to find the minimum error value. On the other hand, if it is too large, the weight update will over-shoot the error minimum and the weights will oscillate or even diverge. Generally one should try a range of different

values between 0.1 and 0.0001. It is not necessary to keep the learning rate fixed throughout the learning process. Sometimes it is also good to choose a learning rate that varies.

As mentioned in previous paragraphs, it is not easy to determine the optimal neural networks. Therefore, we need to evaluate the performance of different configuration of neural networks in order to determine the optimal neural networks. The question is how can we evaluate the performance of neural networks. A portion of a training data set can be used to optimise neural network training procedure [15]. The data set withheld from the network training is called the validation data set.

We can use a validation data set to choose the best neural network. Firstly, we need to split the available data into training and validation sets. Secondly, we train various neural networks using the training set. Thirdly, we test each one on the validation set and finally, we choose the neural network which performs best on the validation data set. Since the availability of a training data set is usually limited, using part of it as a validation set is not practical. An alternative is to use a k-fold cross-validation.

The procedure for performing k-fold cross-validation is as follows [15]:

1. Divide randomly the set of training data into k distinct subsets.
2. Train the network using k-1 subsets.
3. Test the network on the remaining subset.
4. The average performance on the k omitted subsets is the estimate of the generalisation performance.

2.2 Unsupervised Learning

In unsupervised learning, neural networks are able to learn from observing data without being told to associate the observations to given desired responses and without even given any hint about the goodness of a given response [51]. Unsupervised learning is also applied in machine learning and artificial intelligence. Generally, we apply unsupervised learning to get new explanation or representation of observation data.

There are two main applications of unsupervised learned neural networks. The first application is pre-processing of raw input data to extract useful and appropriate features. This task is very important before classification and regression are performed. It is also called dimensionality reduction or feature extraction. With reduced set of input variables the input-output mapping done for function approximation or classification becomes simpler and less training samples are required. The second application is clustering or grouping of unlabelled training patterns based on competitive learning.

Competitive learning is the most popular unsupervised learning technique. It is described by the following rules [15]:

1. The neurons in the network are all the same and they respond differently to a given set of input patterns.

2. A specific limit is imposed on the strength of each neuron in the network. A typical example is normalising each weight vector.

3. The neurons compete with each other in accordance with a prescribed rule for the right to respond to a given subset of inputs; consequently only one

output neuron is active at a time.

4. The neuron that wins the competition is called a winner-takes-all neuron.

5. The individual neurons of the network assume the role of feature detectors for different classes of input patterns.

Self-organising map and snap-drift modal learning networks are examples of unsupervised neural networks for grouping or clustering unlabelled input data. They are described in detail in the following sections.

2.2.1 Self-Organising Maps

Self-organising maps(SOMs) are unsupervised trained neural networks where their output neurons organise themselves based on competitive learning. The main purpose of a SOM is to transform an incoming signal pattern of arbitrary dimension into a one or two dimensional discrete map, and to perform this transformation adaptively in a topologically ordered fashion [38]. Kohonen network is a type of SOM which has a feed-forward structure with a single computational layer arranged in rows and columns [38]. Each neuron is fully connected to all the source nodes in the input layer. The architecture of Kohonen network is depicted in figure 2.1.

The algorithm for training SOMs has four major components [29]:

1. Initialising the synaptic weights in the network by assigning them small values picked randomly.

2. Competition: for each input pattern, the neurons in the network compute their respective values of a discriminant function such as Euclidean distance

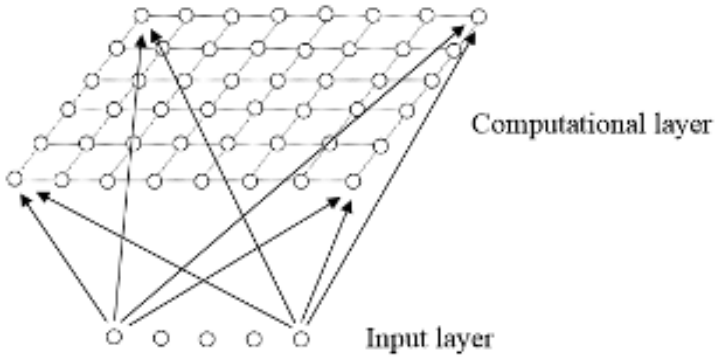


Figure 2.1: Topology of SOMs [38]

from input pattern. This discriminant function provides the basis for competition among the neurons. The particular neuron with the largest value of discriminant function is declared winner of the competition.

3. Cooperation: the winning neuron determines the spatial location of a topological neighbourhood of excited neurons, thereby providing the basis for cooperation among such neighbouring neurons.

4. Weight adaptation: excited neurons increase their individual values of the discriminant function in relation to the input pattern through suitable adjustments applied to their synaptic weights. The adjustments made are such that the response of the winning neuron to the subsequent application of similar input pattern is enhanced.

The detail of the algorithm for training SOMs is summarised as follows [29]:

1. Initialization: choose random values for the initial weight vectors $W_j(0)$.

2. Sampling: draw a sample x from the input space with a certain probability.

3. Similarity matching: find the best matching (winning) neuron $i(x)$ at time step n using the minimum distance criterion.

$$i(x) = argmin_j \|x(n) - w_j\| \tag{2.2}$$

4. Updating: adjusting the synaptic weight vectors of all excited neurons using the update formula.

$$w_j(n+1) = w_j(n) + \eta(n) h_{j,i(x)}(n)(x(n) - w_j(n)) \tag{2.3}$$

   Where: $\eta(n)$ is the learning rate parameter and $h_{j,i(x)}(n)$ is the neighbourhood function centred around the winning neuron $i(x)$.

5. Continuation: continue with step 2 until no noticeable changes in the feature map are observed.

Even though an implementation of the above algorithm for training SOMs is straight forward, there are some practical challenges when we apply SOMs to perform unsupervised learning tasks. For most of them, there are hints recommended based on previous experiments [38]. To initialise the weight vectors of output neurons, there are two options. The first one is to assign each weight vector with a small random number and the other option is to initialise the weight vectors by a randomly selected input patterns. Usually a very large number of iterations are required to have a good output result. It is recommended that the number of iterations should be at least 500 times the number of output neurons.

The learning time can be divided into two stages, which are ordering and fine-tuning. During the first 1000 iterations, the output neurons are ordered and the remaining iterations are required for fine tuning the output map. The learning rate should be initialised close to one and allowed to decrease linearly or exponentially during the ordering phase and it should be very small of the order of 0.01 during the fine tuning phase.

In addition to the above challenges, determining the size of the neighbourhood of the winning neuron output and specifying how to update each weight vector of neighbour neuron are also other challenges. The simplest way is to choose a rectangular or hexagonal shape neighbourhood centred at the winning neuron and specifying the width of the neighbourhood by a distance between the winning node and the furthest neighbour neuron in the output space. Once the neighbourhood is determined, only the weight vectors of all neurons inside the neighbourhood are updated. It is recommended that a Gaussian neighbourhood function should be considered to ensure that neighbour neurons spatially close to the winner neuron are adapted more strongly than neighbour neurons located further away. Initially during the ordering phase, it is recommended that the radius of the neighbourhood to be more than half the diameter of the output map and should decrease linearly until it reaches one unit. During the fine-tuning phase, the radius of the neighbourhood should be fixed to one unit to include only the nearest neighbour neurons.

A Gaussian function is defined as follows.

$$h_{cj}(t) = h_0 e^{-r_i - r_c{}^2/2\sigma^2(t)} \tag{2.4}$$

Where h_0 is the maximum height of the Gaussian function; $r_i - r_c$ denotes the distance between winning neuron c and neighbour neuron i within the output grid and σ is a time varying parameter that specifies the width of the neighbourhood.

SOMs have been applied successfully in a wide range of applications. Generally, we can categorise all applications into three, which are clustering, visualization and abstraction method [38]. One of the main applications of SOMs is text or document organisation, which enables users to explore, search and filter a huge amount of information [20; 23; 37; 41; 46]. For example, SOMs have been successfully applied to organise documents gathered from Usenet discussions [37; 41],manual pages of C++ class library [46] and document archive comprising articles from a daily Austrian newspaper [23]. SOMs can also be applied for analysing bank customers by identifying groups of customers based on their banking behaviours [32].

One of the limitations of SOMs applied for clustering purposes is the fact that output maps do not show cluster boundaries [46]. Many techniques such as U-matrix display, cluster connections and automatic colouring of output map have been proposed to show cluster boundaries [46]. The other limitation of SOMs with a single two-dimensional output map is their inability to detect the hierarchical structure inherent in document collections. Two methods are developed to address this limitation [23]. These are hierarchical feature map and the growing hierarchical self-organising map (GHSOM). In both methods, a layered architecture that consists of independent SOMs within each layer is used to visualise hierarchical data structure. However, they differ in the way they specify the number of layers and the size of the maps within each layer. In the hierarchical feature map, the number of layers and the size of the maps within each layer

are specified in advance, however, the structure of the hierarchy is determined dynamically to resemble the structure of the input data in the case of the growing hierarchical self-organising map method.

Since SOMs are not effective for pattern classification, vector quantisation technique is incorporated with SOMs to produce a supervised version, which is called learning vector quantization (LVQ) [38]. LVQ starts from a trained SOM with input vectors x and weight vectors w_j and uses the classification labels of the inputs to find the best classification label for each w_j. LVQ has been applied successfully in many real life problems. According to [38], LVQ has performed better than Bayes classifier and K-nearest neighbour methods for speech recognition.

The LVQ uses the following algorithm to move w_j appropriately [38]:

1. If the input x and the associated Voronoi/ weight vector $w_{i(x)}$ (the weight of the winning output node $i(x)$) have the same class label, then move them closer together by

$$\Delta w_{i(x)}(t) = \alpha(t)(x - w_{i(x)}(t)) \tag{2.5}$$

2. If the input x and associated Voronoi/ weight vector $w_{i(x)}$ have different class labels, then move them apart by

$$\Delta w_{i(x)}(t) = -\alpha(t)(x - w_{i(x)}(t)) \tag{2.6}$$

3. Voronoi/ weight vectors w_j corresponding to other input regions are left unchanged with

$$\Delta w_j(t) = 0 \tag{2.7}$$

$\alpha(t)$ is a learning rate and its value is between 0 and 1. It is recommended that its value should be 0.01 or 0.02 initially and let it decrease until it is close to 0 during the final iterations [38]. The reason for starting with a very small learning rate is the fact that the learning vector quantisation is a fine-tuning stage initialised by a trained SOM [38].

2.2.2 Modal Learning Neural networks

Modal learning neural network combines several modes of learning within a single neural network or module in order to achieve learning results that no single mode could achieve through exploitation of the complementary nature of each mode [53]. It is different from hybrid or modular neural networks in which the different learning modes are applied at different modules and/or at separate times. Snap-drift neural network(SDNN) and adaptive function neural network are examples of modal learning neural networks. In adaptive function neural networks, both weight vectors and shape of activation functions are updated simultaneously [36].

SDNN is a simple modal learning method, which swaps periodically between snap and drift learning modes. SDNN was first conceived as a learning algorithm as an attempt to overcome the limitations of adaptive resonance theory(ART) learning in non-stationary environments where self-organization needs to take account of periodic or occasional performance feedback [42]. Snap is a logical intersection learning while drift is a learning vector quantisation. They provide complementary features. Snap captures common elements of group of patterns represented by the minimum values on each input pattern and it contributes to rapid convergence whereas drift captures the average values of the group of

patterns.

The architecture of SDNN consists of three layers as shown in figure 2.2. They are an input layer, a distributed d layer for feature extraction and a selection s layer for feature classification. The distributed d layer groups the input patterns according to their features using snap-drift training algorithm. The D most activated(winning) nodes out of the d layer whose weight vectors best match the current input pattern are used as the input data to the selection s layer. In the s layer, a quality assurance threshold is applied. If the net input of the most active s node is above the threshold, that s node is accepted as the winner and defines the category of the input pattern; otherwise a new uncommitted output node is recruited as the winner.

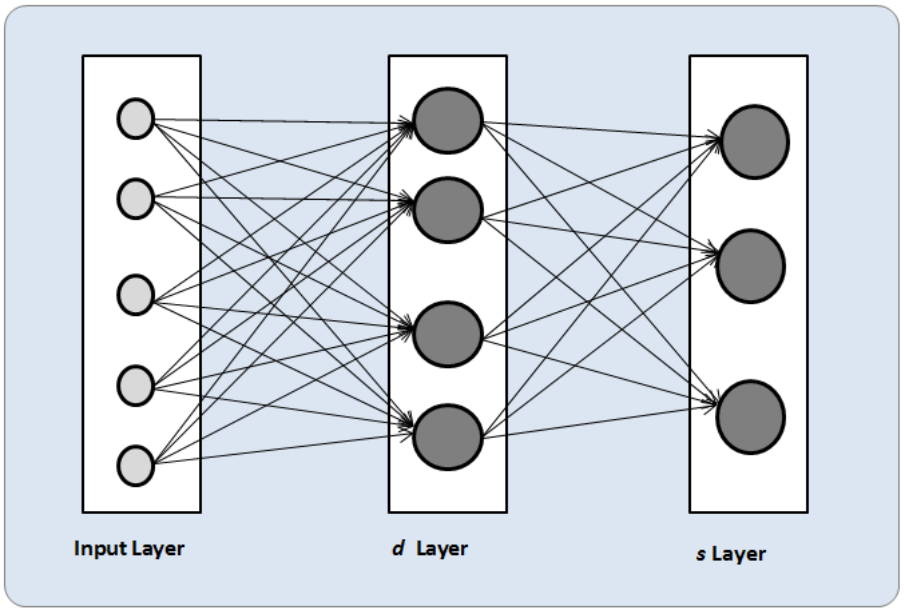


Figure 2.2: Architecture of SDNN

The weight vectors of the D most activated(winning) nodes and the winner of the s layer are updated according to the following equations.

$$Snap - drift = \alpha(Snap) + (1 - \alpha)(drift) \tag{2.8}$$

The above equation defines how the snap and drift learning modes are combined in SDNNs. In successive learning epochs, the learning is toggled between snap and drift learning modes. When α is one, a snap learning is invoked and α is set to zero to invoke drift learning mode. The above weight update equation is further elaborated in the following equation.

$$w_{ji}^{(new)} = \alpha(I \bigcap w_{ji}^{(old)}) + (1 - \alpha)(w_{ji}^{(old)} + \beta(I - w_{ji}^{(old)})) \tag{2.9}$$

Where w_{ji} is a weight vector of either d or s layers, I is a binary input vector, and β is the drift learning rate. After each weight update, the weight vectors are normalized to a unit length.

The objective of the snap learning mode is to find the minimum between the binary input vector and the weight vector of the winning node whereas the objective of the drift learning mode is to minimise the total squared distance between the winning nodes and their closest input patterns.

The mathematical formulation of the objective function of the snap-drift learning algorithm is as follows.

$$Min((I, W) + \|I - w\|) \tag{2.10}$$

Snap-drift learning algorithm has been applied as reinforcement, classifier and unsupervised version to solve various real life problems successfully. For instance, the unsupervised snap-drift has been used in feature discovery and clustering of

speech waveforms from non-stammering and stammering speakers [42], and for classifying user requests in an active computer network simulation environment where the system was able to discover alternative solutions in response to varying performance requirements [24].

2.3 Student Modeling

Intelligent tutoring system is defined as any system which is capable of emulating an instructor's behavior in all aspects relating to supporting students as they acquire knowledge [40]. The key feature of ITS is their ability to provide a user-adapted presentation of the teaching material which is accomplished using artificial intelligence techniques to represent the pedagogical decisions and the information regarding each student.

Most intelligent tutoring systems have four common major components: knowledge domain, student model, teaching strategies and user interface [50]. Knowledge domain stores learning materials that the students are required to study for a topic or curriculum being taught. Student model stores information that is specific to each individual learner and enables the system to identify different users. The teaching strategies component refers to instructional techniques for teaching. For example, the component decides when to present a new topic, how to provide recommendations and guidance and which topic to present. User interface component decides how the system interacts with a user.

Student model is an essential component of an intelligent e-learning environment that enables personalized and adaptive learning. It can contain different kinds of information such as student plans, solution paths, student performance,

preferences, interests, knowledge level, problem solving skills and constraints that the student has violated [28]. The student model in intelligent tutoring systems can be used for different purposes: to determine if the student is ready to continue with the next curriculum topic; to generate explanations according to the student knowledge; to generate problems according to the student knowledge level; and to generate appropriate teaching strategy according to the student knowledge and learning style [28].

There are three approaches to construct student models [27]. The first approach uses a specially prepared task-model pairings. The second approach constructs a student model by mapping behaviour to predefined set of bugs. The third approach infers a student model from observed behaviour. The first two approaches depend heavily on catalogues of mal-rules, which is a simple perturbation of some correct rules collected through an extensive protocol analysis of the domain. In the third approach, the idea is to use a smaller amount of initial knowledge to infer a student model and it is possible by applying machine learning techniques. An emphasis is given to the third approach which applies machine learning techniques. Several machine learning approaches that model students' knowledge status, students' learning style and cognitive style of students are described in the following sections.

2.3.1 Knowledge Status

A hybrid algorithm based on fuzzy-ART2 neural network and Hidden Markov model was applied in order to categorise students' knowledge status into six levels: excellent, very good, good, fair, weak and very weak based on five parameters

collected while students interact with an e-learning system [30]. The five parameters are number of correct answers, number of incorrect answers, time spent to solve a question, time spent reading or interacting with a specific concept and number of attempts to answer a question. The purpose of the student model is to identify whether a student is novice or advanced and choose an appropriate instruction type. The technique has not been evaluated in real educational situations.

A Bayesian network approach to model the skill levels of a student on a learning object was proposed in [44]. The skill levels are categorised into novice, beginner, intermediate, advanced and expert. Students are provided with quizzes on the learning object and the following data are then collected: the number of correctly answered questions, number of incorrectly answered questions, and time taken in answering each question. The modeling result is used by other systems in the learning environment for adaptive selection and presentation of learning material for individualised learning.

A supervised Kohonen network with a hexagonal lattice structure was proposed to classify the knowledge level of a student into 3 categories: beginner, intermediate and advanced [68]. Two student profiles are used to build the student model. They are the implicit and explicit techniques. Explicit data is information about a student which cane be recorded using a questionnaire or registration form. Implicit approach is an alternative of the first approach that analyses the student navigation behavior while using the system. Students usually do not realise that every movement and activities have been captured and recorded in a log file or can be stored in the system database. There are 4 attributes identified to represent the possible student features. The attributes are learning time,

number of backtracking, number of getting help and assessment score. The Kohonen network was trained using a simulated data since real student data was not available. The developed student model can be used in intelligent tutoring systems to adapt the learning sequences to match with students' knowledge level.

A neuro-fuzzy approach was also proposed to model students' behavior that defines knowledge level, mistakes, misconceptions, learning speed, attention and memory limitations [59]. The inputs of the system are students' responses from a set of questions and exercises, the time spent to read a given theory and find correct answers, the number of attempts to find correct answers and the number of times needed to review a given theory. In addition to these input parameters, another parameter, which is expected mean value estimated by human expert is also used. The student model can be used for deciding appropriate teaching strategy. The output of the system is categories for each characteristic: knowledge level, mistakes, misconceptions, learning speed, attention and memory limitations. For instance, for the learning speed, the possible values are slow, rather slow, normal, almost fast and fast. The technique was evaluated using a population of 300 simulated student cases to compare its performance with the decisions of five teachers. The result of the experiment revealed that the overall average classification was 95 percent.

2.3.2 Learning Style

A learning style is defined as an educational condition in which a student is most likely to learn [60]. There are seven types of learning style: visual, aural, verbal, physical, logical, social,and solitary [2]. Intelligent tutoring systems have been

applied machine learning techniques to predict the learning style of a student in order to choose an appropriate teaching method. A dynamic Bayesian network technique was proposed to produce a student model that can be used to filter and sort learning objects according to the students' learning style and preferences [16]. First an initial model is constructed using the index of learning style questionnaire proposed by Felder and Soloman. The model is then refined by applying dynamic bayesian network technique while the student interacts to choose learning objects. The technique has not been evaluated using either simulated or real students.

A student model based on decision tree and Bayesian Markov chain was proposed in [43]. The decision tree was used to classify students into three learning types: challenging, reinforcement, and regular during the beginning of a tutoring session based on students' responses to a few learning related questions. The decision tree was trained with past data labelled manually by teachers. Clustering technique of Bayesian Markov chain was used to model student behaviour into three learning types using data collected as the students interact with the system. The quality of the two models was compared and it was found that the model generated using the clustering approach represent a more distinct set of student learning styles and more homogeneous groups.

2.3.3 Cognitive Style

Cognitive styles are defined as the manner in which students process information [56]. They are different from learning styles, which determine how a student interacts with or responds to new information. A student modeling approach was proposed to identify automatically the cognitive style of students based on their

learning patterns [19]. A k-means and decision tree techniques were applied. The k-means was used to identify similar student learning patterns which was defined by eight attributes: the total number of pages each student browsed, the total number of visited pages, the total number of times each navigational tool was used and the number of repeated visits the students made. The decision tree was used to produce rules that determine the relationship between student learning patterns and cognitive styles.

2.4 Computer-Based Formative Assessments

Assessment is a process to measure the knowledge, understanding, abilities or skills of an individual [54]. We can classify assessments into two main types depending on their purpose [14; 18]. Formative or assessment for learning is an assessment whose purpose is to provide feedback to students in order to enhance their learning. The second type of assessment is summative and its purpose is to provide a mark or grade that measures the performance of students.

Formative assessment is one of the most crucial processes that enhances the effectiveness of learning experiences [12]. Students can benefit from doing self-assessments to test their understanding by trying out things and receiving feedback [12]. Instructors, on the other hand, can identify specific student misunderstandings, provide feedback to students to help them correct their errors, and identify and implement instructional correctives [18]. Moreover, instructors can use the result of the assessment to monitor the performance of individual students so that they can identify students who need more help.

Feedback is the most essential part of formative assessment as the sole purpose

of performing formative assessment is to provide feedback based on students' responses observed from assessments [7]. Feedback is defined as an information communicated to a learner that is intended to modify the learner's thinking or behaviour for the purpose of improving learning [58]. Feedback is concerned with current performances, that is why a new concept known as feed forward is introduced to deal with future performances [1]. Feed forward focuses on how students improve on their follow up assignments. A formative assessment with no feedback has no effect on improving student learning experience. It is also true that a formative assessment with feedback does not necessarily improve student learning performance [10]. For example, if answers are included as part of feedback, the effect of feedback on student learning will be negative [10; 13]. The quality of feedback determines the effectiveness of formative assessment [10].

Several research studies have been conducted to find out what characteristics of feedback actually improve student learning. Most researchers agree that for feedback to be effective, it should be non-evaluative, supportive, timely and specific and should include the comparison of actual performance with an established standard of performance [58]. Researchers have also reported that the content of effective feedback should contain both verification and elaboration [58]. Verification indicates whether student's work is correct or not, while elaboration is an information that provides details of how to improve an answer [58]. The elaboration aspect of feedback can be more specific and directive when it addresses a topic, a response or a particular error. It can also be more general and facilitative when it provides worked examples or gives gentle guidance.

For a formative assessment to be useful it should be performed on a regular basis [7] and tutors should provide both timely and informative feedback

[14]. However, due to large numbers of students and practical constraints such as time and workload pressure, it is often difficult and time consuming for tutors to conduct a continuous formative assessment and provide timely and informative feedback [14]. That is why computer-based assessments were introduced to increase the efficiency and effectiveness of assessments.

Computer-based assessment or e-assessment is the use of information and communication technology (ICT) to support an assessment process [69]. ICT can be used to support a wide range of assessment activities such as designing or setting up an assessment, delivering an assessment, gathering and analysing student responses, marking and giving feedback to students. E-assessment has an advantage over traditional paper-based assessment due to the capabilities of ICT [21]. One of the characteristics of ICT, which makes e-assessments effective in gathering student responses and delivering feedback to students quickly, is speed. The second characteristic of ICT is storage capacity that is its ability to store large amount of questions, student responses and pre-defined feedback. The third characteristic is the capabilities of ICT to analyse student responses, automate feedback process and adapt assessments to meet different student needs. The fourth characteristic is communication, which enables students to access and perform an assessment task anytime and anywhere and to collaborate with peer students from different locations to work on group assessment tasks. Finally, ICT enables design of interactive assessment tasks using different media formats such as texts, audio, images and videos.

Computer-based assessments can be categorized into three : those which are based closely on existing paper-based assessments; those that use new formats including multimedia, constructed response, automatic item generation and au-

tomatic scoring; and those that use complex simulations, artificial intelligence, statistical techniques and cognitive science [52]. We can also classify computer-based assessments into summative and formative depending on their purpose similar to how traditional paper-based assessments are classified.

Formative computer-based assessment is a set of processes involving ICT in order to gather evidence about learner's state of understanding relative to desirable goals so that individuals are enabled to take actions which bring about changes in learners' skills, knowledge and understanding, or in teachers' pedagogical practice [21]. Instructors can use the evidence about learner's state of understanding relative to desirable goals to correct their instructional strategies and generate feedback to improve student learning. Feedback is the most important component of formative e-assessment as it is for formative paper-based assessments for the same reason mentioned previously.

A brief review of computer-based formative assessments is presented in the following sections. The review focuses on how feedback process of formative assessments is supported or automated using ICT for three different assessment tasks, which are multiple choice questions (MCQs), short free text responses and problem solving exercises. For formative computer-based assessments, marking or scoring is less important, as their purpose is to improve student learning via feedback. That is why the focus is on feedback process not on automating marking.

2.4.1 Multiple Choice Questions

A multiple choice question (MCQ) is a question in which students are asked to select one alternative from a given list of alternatives in response to a question stem [17]. MCQs are comprised of four parts, which are stem, options, key, and distractors [47]. Stem is the text of the question. Options are the choices provided after the stem. Key is the correct answer in the list of options where as distractors are the incorrect answers in the list of options.

MCQs are the most used assessment type in higher education institutions due to its suitability in environments with a large number of students and reduced resources [49]. One of the most important features of MCQs is the fact that they can be easily marked and the score can be both accurate and objective [17]. Another important characteristic of MCQs is its capability to assess the different cognitive levels. For example, a question may simply challenge a student's ability to recall facts, while another may test a student's ability to understand a concept, principle or procedure; or, at a higher level, a question may test a student's ability to evaluate given information [17]. Furthermore, MCQs can be designed with a diagnostic end in mind, in order to find out whether specific areas of a given subject are adequately known or understood, or in order to detect misconceptions [17].

Several formative computer-based assessment tools exist that support MCQs assessment task. Most of them do not provide feedback other than automated score results. Only tools that attempt to provide elaborated feedback are considered. They are described in the following sections.

2.4.1.1 GAM-WATA

GAM-WATA is a multiple choice question web-based quiz-game-like formative assessment tool [67]. Its main function is to help teachers administer web-based formative assessment and interact with students in order to improve student-learning effectiveness and encourage students to perform self-assessment spontaneously. This tool implements three strategies, which are repeat the test, correct answers are not given and all pass and then reward. Students are allowed to repeat a test until all question items are answered correctly. Every time a student tries a test, correct answers are not given, instead an item question is removed from the test if the student answers it correctly three times in succession. A student gets a reward when he/she answers all questions correctly three times in a row.

The tool provides two kinds of feedback to students. The first one consists of an immediate on-line hint given to students while they are taking a test. This feedback is optional and can be triggered by students when they feel an item question is difficult. To reduce the difficulty of the question, students can choose either to eliminate one of the incorrect options or to get information about how their peers answer the question. A teacher provides the second type of feedback when students send their questions by email.

The tool was evaluated and compared against a web-based formative assessment and paper-and-pencil based test in an elementary school environment [67]. It was concluded that the students in the GAM-WATA group participate more actively in web-based formative assessment than students in the normal web-based formative assessment group and GAM-WATA improves learning effective-

ness more than the other two approaches. It was argued that the challenge and game mechanism included in the GAM-WATA tool promote the motivation of students to actively participate in web-based formative assessment and suggested that more effective strategies should be incorporated in web-based formative assessment to construct a successful e-Learning environment.

One of the limitations of the tool was the fact that the feedback provided by the tool is not as direct and resourceful as the reference information provided by teachers to facilitate student learning in a traditional learning environment [67]. It was recommended that techniques of intelligent tutoring should be integrated into the tool to provide feedback with high informativeness.

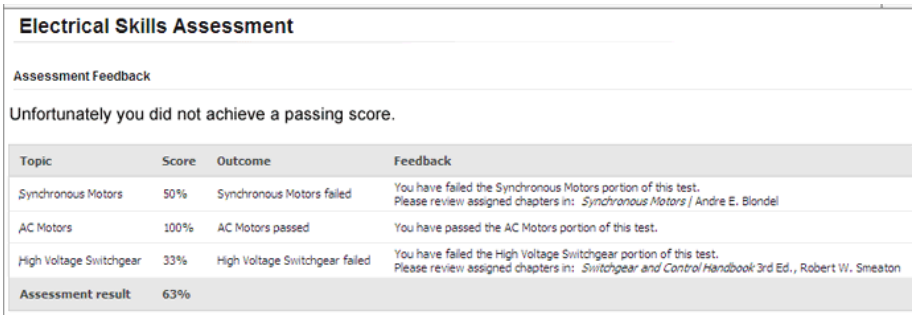
2.4.1.2 Question-mark Perception and Hot Potatoes

Hot Potatoes is a commercial web-based software tool that enables the creation of interactive MCQs, short answer, jumbled sentence, cross-word, matching and gap-fill exercises [31]. The tool provides useful and targeted feedback based on students' response. The feedback is written when MCQs are prepared. Specific feedback is written for each distractor of a given question and additional feedback for a correct option. The feedback for a correct answer is required to help students understand why the answer is correct.

Question-mark Perception is also a commercial web-based software tool that enables educators or trainers to author, schedule, deliver and report surveys, quizzes and exams [55]. It can be used for formative, diagnostic and summative purposes. The item level feedback provided for MCQs is similar to the one provided by the Hot Potatoes tool. In addition to item level feedback, Question-mark Perception provides topic and assessment level feedback. Item level feedback

provides a specific diagnosis that correct misconceptions related to a particular question [55]. The disadvantage of too much item level feedback according to the developers of Question-mark Perception tool is that it can encourage students to get the question right instead of understanding it. That is why Question-mark Perception offers topic level feedback based on aggregated scores of items within a specific topic or sub-topic.

The topic level feedback provides two types of feedback; one for an aggregated score below fifty percent and another for aggregated score above fifty percent. The content of the topic level feedback is not specific and elaborated as shown in figure 2.3. It informs students whether they pass or not. If they fail, students are asked to re-read relevant chapters. Considering the content of the topic level feedback, it is difficult to agree with the developers of Question-mark Perception who argue that the topic level feedback can diagnose knowledge and skill deficiencies, correct misconceptions and prescribe a learning event that would help participants improve their learning performance.



Electrical Skills Assessment

Assessment Feedback

Unfortunately you did not achieve a passing score.

Topic	Score	Outcome	Feedback
Synchronous Motors	50%	Synchronous Motors failed	You have failed the Synchronous Motors portion of this test. Please review assigned chapters in: *Synchronous Motors* / Andre E. Blondel
AC Motors	100%	AC Motors passed	You have passed the AC Motors portion of this test.
High Voltage Switchgear	33%	High Voltage Switchgear failed	You have failed the High Voltage Switchgear portion of this test. Please review assigned chapters in: *Switchgear and Control Handbook* 3rd Ed., Robert W. Smeaton
Assessment result	63%		

Figure 2.3: Topic level feedback from Questionmark Perception tool[55]

2.4.2 Short-free-text Responses

Short-free-text response questions require students to construct responses freely in natural language without the need to select from options of answers. Short-free-text responses are usually a few phrases or three to four sentences [33]. Current marking algorithms can not handle long student responses, as a result short-free-text responses are restricted to no more than 20 words [33; 34]. Moreover, the current tools have difficulty in marking student responses that include both aspects of correct and incorrect responses [34].

Most computer-based assessment tools that support short-free-text responses assessment tasks focus on automating marking rather than on providing feedback for formative purposes. The widely used software tools for automating marking of short-free-text responses are C-rater developed by Educational Testing Service and auto-marking tool based on information extraction and machine learning techniques developed at the University of Cambridge [61; 62]. They provide feedback that includes either score result or the correct answer when students' response is incorrect or incomplete.

Basically, there are two methods of marking free-text responses automatically. These are knowledge based systems or machine learning techniques [35; 62]. The knowledge based systems are more accurate and require less training data than machine learning techniques. However, writing patterns for knowledge based systems are very tedious work and they require expertise in both domain of examination and computational linguistics.

In the knowledge based systems, experts in the domain of assessment tasks who are instructors or tutors list possible correct answers per question from their

experience or past student responses. The next challenge is to represent the possible correct answers as a pattern that could store all the variable information using predefined grammar. Patterns are written usually by hand and contain recurring head words or phrases which are annotated using part-of-speech tags such as noun phrases and verb groups. To mark a new student response, firstly, it has to be processed using natural language processing technique and then represented as a pattern in the same way as the correct answers are represented. Secondly, the pattern for the new student response is matched with the pattern of the model answer and a marking rule is applied to calculate a score. Most of the time simple marking rule that specifies pass or fail criterion is used. In this rule, a full mark is given if there is exact match between the new student response and model answer patterns, otherwise zero mark is given. Other marking rules that award scores between zero and full marks can be used for partial patterns match.

Supervised machine learning approaches such as decision tree learning and Bayesian learning were applied to automatically classify short-free-text responses to facilitate automated marking [62]. Previous student responses are used as training and testing data sets. Since supervised learning is applied, the first task is to label manually each previous student response a score ranging from zero to full-marks by a domain expert. For a machine learning algorithm to process student responses, they have to be represented as a set of attributes where words of a student response are considered as attributes. When all the words of a student response are considered as attributes, it is called non-annotated data. Alternatively, only parts of a student response that are relevant could be considered as attributes, in which case it is known as annotated data. Once the

past student responses are represented as a set of attributes and then labelled, a machine learning technique can be applied to build an abstract model to represent the training data which could be used to predict the score of a new student response.

Even though most developed computer-based systems focus on automated marking of short-free-text response questions, there are a few assessment tools that concentrate on both automated marking and immediate delivery of formative feedback. A tool developed by intelligent assessment technologies (IAT), which has been deployed by the UK Open University, is an example of such assessment tools [35]. It was developed based on natural language processing techniques and information extraction techniques, and provides an authoring tool that can be used by a question author with no skill in natural language processing [35]. The authoring tool allows question authors to focus on writing model answers for a question and the keywords for each model answer by hiding the complexities of natural language processing [35]. The tool also provides a marking engine that performs a match between student responses and predefined model answers.

The tool provides instantaneous feedback in three stages where students are allowed to attempt three times only before they can receive the model answer for a particular question [35]. The system also provides other predefined feedback associated with possible incorrect responses specified by a question author. If a student response is incorrect then it is matched with the predefined possible incorrect responses and a predefined feedback is provided when there is a match with one of the possible incorrect responses, otherwise, the student is provided with feedback that informs that his/her response appears to be incorrect or incomplete.

If we look at figure 2.4 that demonstrates the three feedback stages, at the first attempt the tool provides feedback that says your answer appears to be incorrect or incomplete in some way. The tool does not give specific feedback that addresses the incorrect response since the response could not match with the possible predefined incorrect responses. The reason why there is no match is the fact that the question author could not anticipate student responses similar to what given in the first attempt. This is one of the weaknesses of the tool. Fortunately, during the second attempt, the tool provides predefined feedback that matches one of the possible predefined incorrect responses. Finally, the student responds correctly for the third attempt and receives feedback that verifies that the answer is correct and contains also a model answer.

2.4.3 Problem Solving Exercises

2.4.3.1 Retina

Retina is a tool that helps instructors and students by observing student's programming activities [48]. The tool can record the compilation attempts, compilation errors and run time errors. It enables students to review their past actions, see how they relate to other students and get suggestions on how to avoid common errors encountered in the previous assignments and recommendations about how to go forward.

It also allows instructors to understand more about what their students are doing. The instructor can analyse the individual performance to detect which students need more tutor support and by analysing the overall performance of the class which topic needs more explanation can be identified.

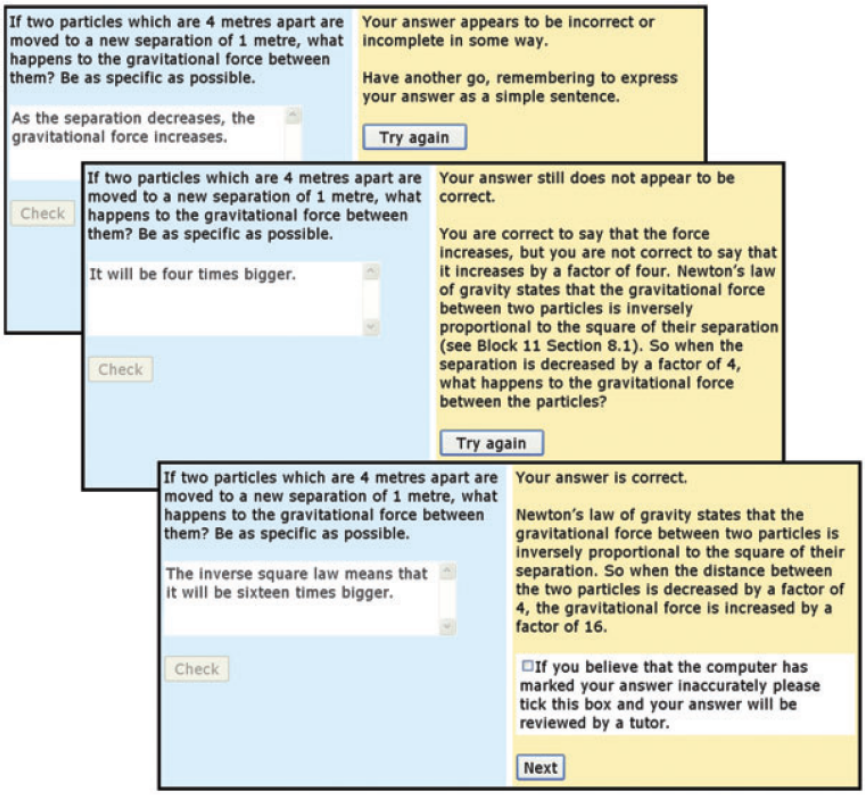


Figure 2.4: Three stage feedback extracted from [35]

The Retina tool adopts two approaches of data analysis. The first approach applies statistical methods to provide summary for the students and instructors and give suggestions to students. The second approach uses rule-based expert system to provide appropriate, immediate and real time recommendations to students while they are doing their programming activity for a particular assignment.

The rule-based expert system consists of three rules:

1. If the rate of errors per compilation is higher than normal of a student, recommend that the student attempts to work in smaller intervals or address compilation errors that are at the top of the list, which may be causing other

errors to appear.

2. If the amount of time spent by a student on an assignment is more than twice the suggested time, recommend that the student is spending too much time on it and should seek the assistance of a member of the teaching staff for help.

3. If the same error occurs on the same line on more than four consecutive compilation attempts, explain that error in simple terms and recommend a possible way to fix it.

The tool was evaluated to determine if the potential aims of enhancing teaching and student learning experience were achieved. Programming activities of 48 students taking a first year course were collected by Retina to conduct the evaluation process. However, only the responses of the instructors of the course were assessed. According to the instructors' experience, the Retina helps them to improve their interaction with the students and increase the quality of the lectures.

Whether the recommendation feature of the Retina tool facilitates the improvement of student performance was not evaluated. This could have been done from the records of the student programming activities held by the Retina tool. We can check if the number of compilation and run time errors and the time spent per compilation attempt gradually decreases as more recommendations are given to the student.

The recommendation feature has applied intelligent data analysis technique using a rule-based knowledge base even though it is a simple knowledge base consisting of three rules. The recommendation is immediate and real time, however,

it provides solutions to each compilation and run time errors.

2.4.3.2 BLASTOFF

BLASTOFF is an individualized interactive formative assessment, which is designed to help instructors to create easily individualized drill-and-practice questions that provides students with interactive feedback to basic accounting problems [11]. The tool is implemented as a spreadsheet template file that contains accounting problems and their corresponding answers and feedback designed by an instructor. It eliminates the burden on instructors to mark and provide feedback on individual questions. Individual questions are created automatically from a template question by choosing a combination of options randomly to reduce student temptation to take inappropriate short-cuts.

Students can receive two types of individualised feedback. The first type of feedback suggests hints to help students start working on a question. The hints have two parts, which are general advice for students that refer them to useful learning resources and the final numerical answers. The second type of feedback is offered to students when they finish attempting a question and it provides a general feedback and assessment of answers. The general feedback contains information for students to refer them to upcoming relevant tests, good spreadsheet practices, or simply other learning resources that the student might use to extend their learning beyond the current concepts being reinforced.

Assessment of answers tells a student whether the attempted answer is correct or not and provides feedback for incorrect answers covering a variety of likely errors. According to the author, the tool has limitations. It does not identify the specific error a student has made for a particular incorrect answer, as a result

customized feedback is not possible. Instead, general feedback that includes a variety of likely errors is offered to students. Even though the tool has been applied in accounting problem exercises, it can be applied to promote learning of other rule-based procedures and concepts.

2.4.3.3 ProtoAPOGEE

ProtoAPOGEE is a prototype for automated project grading and instant feedback system in teaching web-based computing for upper division level students, where advanced issues such as atomicity of database access, thread safety, reliability, robustness and security are a major concern [26]. This tool can evaluate graphical user interface (GUI) programs, unlike previous assessment tools that can only work with text mode programs, based on open-source web application testing tools such as Ruby. However, it cannot check how the solution is implemented that is its quality and style. The tool checks only whether the functional requirements are met by inducing test cases for each requirement.

ProtoAPOGEE comprises three main modules. The first one is a project specification tool that allows instructors to specify the background information, requirements, grading policy, and test cases for evaluating requirements. The second module is an automatic grading, which drives an Internet browser at the background and evaluates a submitted student project using stored test scripts. The last module deals with grade report, which includes an itemized grading summary on all requirements pre-set by an instructor, step-by-step animation playback of the evaluation of each requirement and informative textual feedback.

ProtoAPOGEE provides feedback information at two levels. These are summary information of a project that includes grade received by a student and

detailed feedback for each requirement and their corresponding test cases. The detailed feedback has two major sources of information, which are detailed analysis of each failed test case and their corresponding hints and guidance information. The hints and guidance information are configured by an instructor or automatically by the tool for each failed test case. The tool can not diagnose why the test case of a particular student fails, as a result it is impossible to give individualized feedback to guide him in correcting his errors or misconceptions. The tool provides general hint or guidance information for each failed test case, which is the same for all students who make the same failed test case.

2.4.3.4 Environment for Learning to Program

Environment for learning to program (ELP) is an on-line, active, collaborative and constructive web environment to support novice programmers in teaching Java, C# and C [64]. The tool supports fill-in the gap programming exercises where an exercise consists of a program with some missing lines of code, which are required to be completed by a student. The gaps usually contain helpful hints describing the missing code.

It does not provide feedback to compilation errors; instead it customizes the compilation error messages in an attempt to reduce the complexity of writing programs. The most important feature of the tool is the program analysis framework that can analyse students' fill-in the gap programming exercises and provides feedback on the quality and correctness of their solution. The framework can perform static and dynamic analysis.

Static analysis is the process of analysing source code without executing it for evaluating the quality of students' programs. It is conducted in two stages, which

are software engineering metrics and structural similarity analysis. The software engineering metrics analysis is performed based on the software complexity metric and good programming practice guidelines. It contains eight functions: program statistics, cyclomatic complexity, unused parameters, redundant logic expression, unused variables, magic numbers, access modifiers and switch statements. Once the software engineering metrics analysis is done, the structural similarity analysis refines its results to compare the structure of the student solution with model solutions in order to identify similarities and differences.

Dynamic analysis involves execution of students' solution through a set of test data to estimate correctness and detect at which gap errors occur using black box and white box tests. Black box testing is carried out by executing students' programs through a set of test data and capturing gap outputs to be sent back to the server to check correctness of the output. The main purpose of the white box testing is to discover any possible logic errors, which are not revealed in the black box testing, or to detect gaps that have logic errors, which lead to incorrect outputs. White box testing inserts a student's gap solutions, one at a time into an automated test framework to compare the outputs of each gap with the outputs produced by the corresponding gap solution.

The tool only works for fairly simple introductory programs and for well-formed gaps such as a statement, block of statements, a method or a complete class. Two evaluations were conducted to discover whether the system makes learning to program easier. The first evaluation was for a laboratory-based introductory Java class in which thirty students participated at the end of week 5 of their course. According to the feedback from the participant the students enjoyed using ELP and agreed that the system makes compiling and writing computer

programs so much easier with the gap type exercises and customised compilation error messages.

2.4.3.5 CourseMarker

CourseMarker is a software tool that supports the automated assessment of Java-based course works [63]. This tool is developed based on previously well known computer-based tool known as Ceilidh. Ceilidh is the first computer-based assessment tool that supports the full life-cycle of assessment of programming coursework [8]. The main component of the CourseMarker is its marking system which uses software metrics for program functionality, complexity, efficiency, style, test data coverage and programming skill in order to automatically mark programming coursework and to provide instant feedback.

The marking system includes generic marking schemes, which can be customized for a particular programming exercise. The marking system is comprised of two types of tools, which are dynamic and static tools. The dynamic tool executes the compiled student solutions against a set of test data in order to assess whether they conform to predefined specifications of an exercise to be marked. This tool provides predefined feedback such as excellent, very good, good, fair, and average and poor based on marking result range values. This type of feedback is not useful for students to identify their programming errors and improve their solution.

The static tool checks the source codes of student solutions to assess how well the source codes are written. This tool consists of different tools such as topography and feature tools, which check the different aspects of the quality of a source code. The topography tool checks the readability and maintainability of

a source code by measuring source code layout, indentation, choice and length of identifiers and usage of comments [63]. Based on the result of the measurement, the topography tool provides a mark and appropriate feedback. The feedback mechanism of the topography tool enables an exercise author to identify marking result ranges and then write an appropriate feedback for each mark range value. For example, if we consider a topography measure that counts the average length of identifiers (AIDL), a list of possible mark ranges and their corresponding feedback depicted in figure 2.5 could be generated by an exercise author.

Test result for AIDL	Example feedback
Less than 2	Your average identifier length is too small
Between 3 and 2	Try increasing the length of your identifiers
Between 3 and 10	Your identifiers have proper length
Between 10 and 15	Try decreasing the length of your identifiers
More than 15	Your average identifier length is too large

Figure 2.5: Feedback given by topography tool extracted [63]

The feature tool examines student's solution for the presence or absence of certain keywords that are specific to a particular exercise [63]. For example, the tool can be configured to check the occurrence of language specific features such as the use of switch statements, correct declaration of methods and variables and use of loops. The feedback mechanism of the feature tool allows exercise authors to write feedback for each specified feature that should be provided to students. The first feedback is written if the feature is present while the second is written if the feature is not found. The feedback that can be provided to students is obviously limited by the fact that the only available information is whether a particular feature is present or not. The feedback, as it is illustrated in figure 2.6, could not be more than informing students about the presence or absence of

a feature.

Example feature	Feedback (if ok)	Feedback (if not ok)
`[dD]ouble`	You are using doubles	Doubles are not used!
`IOException`	Exceptions are used	No usage of exceptions!
`final.*static==1`	1 constant found	No constants declaration!
`switch==1`	1 switch statement found	No switch statement found!
`case>=10`	At least 10 case found	Too few case statements!
`if.*else==0`	OK, no if/else found	You should not use if/else!
`while==2`	2 while loops found	2 while loops are missing...
`(for|while)`	Loop construct found	You are not using loops!
`getChar.*\(.*\)`	Method getChar() called	Method getChar() not called
`close.*\(.*\)==2`	The 2 files are closed	2 files are left open on exit!
`super.add\(.*\)`	"super" call is correct	Incorrect usage of "super"

Figure 2.6: Feedback given by feature tool extracted from [63]

2.4.4 Summary

Several computer-based formative assessments have been described in the above sections. We can characterise computer-based formative assessments by the type of the assessment they support and how feedback process is supported. Some of the most common types of assessment are multiple choice questions, short-free-text answers, fill-in the gap exercise and problem solving exercises.

The various assessment types are appropriate to assess different learning objectives and they pose different challenges when we attempt to automate their assessment processes. The multiple choice questions, for example, can assess effectively low level learning objectives such as student's ability in remembering and understanding concepts of a given topic. It is the most suitable assessment type for automatic scoring.

There are three types of feedback. The simplest form of feedback is to inform a student whether his/her response is correct or not and to tell the student the

correct answer if the response is not correct. The feedback is usually tied to each individual question item in the case of multiple choice questions and short-free-text responses assessment forms. In the case of a problem solving exercise such as programming, the feedback is tied to each test case or feature extracted from static analysis of a source code submitted as part of a response to a programming exercise.

The second type of feedback does not tell the correct answer if a student responds incorrectly. Instead, students are encouraged to attempt again. This form of feedback can be done in two ways. Firstly, students are told that their response is not correct and are encouraged to attempt again until they answer it correctly. Secondly, students are told why each response is not correct and a hint or guide on how to improve his/her response on the following attempt is provided. The challenge for this form of feedback is to list all possible incorrect responses. As it is mentioned in previous sections, experts on domain knowledge and past student responses are used to extract possible incorrect responses. Once possible incorrect responses are listed, appropriate feedback for each of them can be constructed.

The last form of feedback is a feedback not tied to individual question or feature. It is sometimes called topic level feedback to differentiate it from item level feedback, which is a feedback tied to individual question or feature. Topic level feedback has been used for multiple choice questions in [55]. The feedback is constructed based on the aggregate score of all multiple choice questions related to a particular topic or sub-topic. The feedback informs students to re-read relevant teaching material if the aggregate score is below fifty percent, which is the same for all failed students. Currently, as the literature review reveals, topic level feedback

has never been used for short free-text responses and problem solving exercises assessment forms. In this research, application of unsupervised learning techniques is investigated for developing effective topic level feedback that improves student learning based on responses observed from multiple choice questions and problem-solving exercise assessment tasks. Multiple choice questions can be used to improve understanding of a topic or subtopic whereas programming exercises can be used to develop programming ability for beginners.

Chapter 3

Unsupervised Multiple Choice-based Student Modelling

This chapter describes the proposed novel method for identifying student group profiles based on student responses to a set of multiple choice questions for the purpose of constructing diagnostic feedback using snap-drift modal learning neural network.

Firstly, we define the learning task that needs to be performed by the snap-drift learning modal learning networks. Once the learning task is defined, a snap-drift learning algorithm is implemented and then training data sets are prepared using two real assessment tasks.

Secondly, a method for representing identified student groups is proposed; analysis of the learning behaviour of the snap-drift modal learning neural networks is described; criteria for determining the usefulness of student group profiles are defined; and student group profiles are identified using the implemented snap-drift modal learning neural networks based on the insight gained into the learning

behaviour and the defined criteria. Once a set of represented student group profiles are identified for the two assessment tasks, the construction of diagnostic feedback for each student group is explained.

Thirdly, how assessment sessions are conducted to gather student responses and the analysis of the gathered student responses are described. The objectives of the analysis are firstly to test if student responses are assigned to their appropriate student groups. Secondly, to assess if student group profiles facilitate the process of identifying gaps of understanding and misconceptions. Thirdly, to assess the impact of diagnostic feedback on student learning performance.

Finally, a summary of the main points of the chapter is presented.

3.1 Learning Task

There are different forms of assessment, which can be used to gather student responses. Multiple choice questions (MCQs) is an assessment type that has been introduced in higher education due to its suitability in current higher education environments with a large number of students and reduced resources [49] and it can be used to assess different cognitive levels. In addition to this, MCQs can be designed with a diagnostic end in mind, in order to find out whether specific areas of a given subject are adequately known or understood, or in order to detect misconceptions [17].

It is not difficult to generate a feedback tied to an individual question. In this case, feedback can be constructed for each option of a given question. However, this type of feedback is not effective to assess the ability of students to understand a topic or sub-topic using only one question. We can only assess student's ability

to recall a particular fact using one question, but we obviously need more than one question to assess student's ability to understand a particular topic. This implies that we should consider different combinations of responses to a set of multiple choice questions to construct effective feedback. For example, if we consider five multiple choice questions with five options (A, B, C, D, E), the possible number of combinations of responses can be calculated as follows:

1. Two-combinations of responses:
$$= \frac{5!}{2!3!} * 5^2 \tag{3.1}$$
2. Three-combinations of responses:
$$= \frac{5!}{3!2!} * 5^3 \tag{3.2}$$
3. Four-combinations of responses:
$$= \frac{5!}{4!1!} * 5^4 \tag{3.3}$$
4. Five-combinations of responses:
$$= 5^5 \tag{3.4}$$

The total possible number of combinations of responses based on the above formula is 7750. The total possible number of combinations of responses increases exponentially as the number of multiple choice questions increases. Therefore, it is not feasible to construct a feedback for each combination of responses. Even though each student is unique, there exists only a limited number of different ways of understanding and range of misconceptions of a topic [3]. Hence, we only need to identify the different groups which are characterised by similar gaps of understanding, level of understanding and/or common misconceptions and assign a student response to an appropriate group automatically.

One possible solution is to analyse manually a collection of responses of students by a human expert in order to categorize the responses into groups of

similar knowledge or understanding levels. Once the groups are identified then each group can be analyzed to characterise its understanding or knowledge level. This solution is very time consuming and it is not practically possible for a large number of student responses.

Another approach is to use neural networks, which are able to derive meaning from complicated and/or imprecise data and to extract patterns that are too complex to be noticed by many other computational techniques [25]. These characteristics make neural networks a powerful method to model human behavior and a useful technique to create user models for hypermedia applications [25]. A neural computing technique, which is snap-drift modal learning neural network is applied to identify the different groups which are characterised by similar gaps of understanding, level of understanding and/or common misconceptions and to assign a given student response to an appropriate group automatically.

The input patterns for the learning task are a set of student responses gathered from assessment sessions. An input pattern (input vector), which is a sequence of responses, is defined mathematically as follows:

$$X = \{x_1, x_2, x_3, ...x_n\} \tag{3.5}$$

Where n is the number of multiple choice questions and

$$x_i \in \{a, b, c, d, e\} \tag{3.6}$$

3.2 Snap-Drift Modal Learning Neural Network

Snap-Drift modal learning neural network (SDNN) is unsupervised learning system that combines two learning modes, which are snap and drift. It is described in detail in chapter two. The focus in this section is to describe how it was implemented in order to perform the specified learning task. The main component of the snap-drift modal learning neural network is the snap-drift learning agent that represents the topology, group allocation map and learning algorithms of snap and drift. Its inputs are learning parameters and training data. It performs learning for a given epoch and returns a group allocation map that contains the index of all training patterns and their corresponding winning nodes of s layer. The algorithm of the snap-drift learning agent is described using a pseudo-code as follows:

1. Get learning parameters (d, s, D ,β_1,β_2 and quality assurance threshold)
2. Get training data.
3. Get current epoch.
4. Create an empty group allocation that maps patterns to winning nodes.
5. Create d and s layers based on the learning parameters and dimension of training patterns.
6. Set an appropriate value of α depending whether the epoch is odd or even.
7. FOR each input pattern of the training data
   (a) Find the D winning nodes at d layer with the largest net inputs.
   (b) Use equation 2.9 to update the weight vectors of the D winning nodes.
   (c) Normalise the updated weight vectors.
   (d) Set the output of the D winning nodes at d layer to 1 and the output of the remaining nodes to 0.
   (e) Consider the outputs of the nodes in the d layer as input patterns to the s layer.

(f) Find a node at the s layer with the largest net input.

(g) IF net input of the node with the largest net input is greater than the quality assurance threshold THEN

   i. Use equation 2.9 to update the weight vector of the winning node and set the winning node as committed.

   ii. Normalise the weight vector of the winning node.

   iii. Add the current input pattern and the winning node to the group allocation map.

(h) ELSE

   i. Select uncommitted node from the nodes of the s layer.

   ii. Use equation 2.9 to update its weight vector and set it as committed.

   iii. Normalise the weight vector of the selected node.

   iv. Add the current input pattern and the selected node to the group allocation map.

(i) END IF

8. END FOR

9. Return group allocation map

The complete algorithm for the snap-drift modal learning neural network is described using flowchart in figure 3.1.

The Snap-Drift modal learning neural network is implemented in Java. To test whether the learning algorithm is implemented correctly and assess its performance in classifying linearly inseparable patterns, iris data set was used. Iris data set contains 3 classes of 50 instances each, where each class refers to a type of iris plant [65]. The first class is linearly separable from the other two classes, while the second class is not linearly separable from the third class.

The data set is labelled, that means, each pattern is labelled as class one, class two, or class three. The implemented snap-drift learning algorithm is unsupervised; hence, the iris training data set is treated as unlabeled data set by

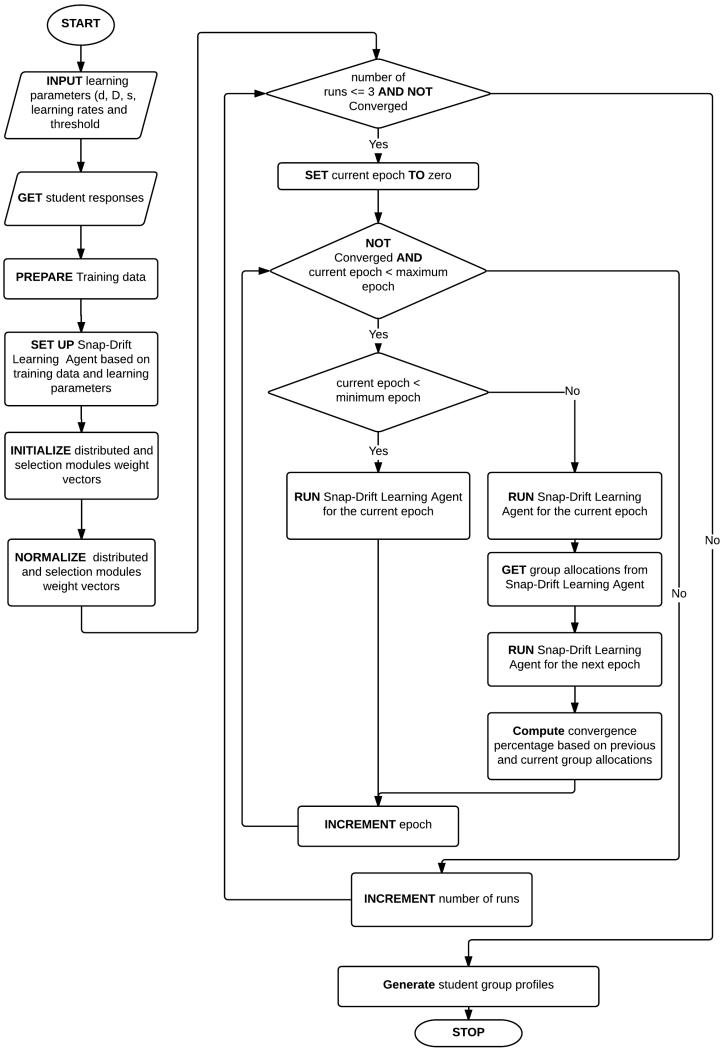


Figure 3.1: Flow chart of Snap-drift modal neural network algorithm

removing class label of all training patterns. However, the class label of each pattern is used to estimate the performance of the implemented SDNN learning algorithm.

We run the implemented SDNN ten times using different combinations of three learning parameters, which are the number of nodes in d layer, the number of selected features or winning nodes (D) and quality assurance threshold. The learning rates were fixed to 0.1 and 0.2 based on previous empirical values, and the number of nodes in s layer was set to 100 so that enough uncommitted nodes are available to allocate to matched training patterns. The weight vectors of the distributed module are initialized by assigning them to a randomly selected training pattern whereas the weight vectors of the selection module are initialized to one based on previous application of SDNN in [53]. The learning stops when the convergence percentage is 95 or when a maximum epoch of 2000 is reached. Convergence happens when each training pattern is mapped to the same winning node for more than two consecutive iterations.

The result of the experiment showed that the minimum and maximum number of groups identified by the implemented SDNN were 2 and 9 respectively. In all runs, training patterns that belong to the first class were not mixed with the training patterns of the second and third classes. This demonstrates that SDNN learning algorithm was implemented correctly and was capable of classifying linearly separable data. Training patterns of the second class were also separated from the training patterns of the third class with a small number of patterns mixed from both classes in all other groups that do not contain training patterns from the first class. The average performance for classifying the non-linearly separable classes, which are the second and third classes, was above 95 percent.

3.3 Data Preparation

Two assessment tasks were chosen to gather student responses. The first assessment task was selected from Introduction to Data Analysis module, which is a core module for first year students at London Metropolitan University. 501 student responses were captured and saved from a web learn where students were allowed to practice multiple choice questions. Five related multiple choice questions that can assess the ability of students in understanding probability topic were only considered. The multiple choice questions are attached in Appendix A.

The second assessment task was selected from Introduction to Programming module, which is also a core module for first year students at London Metropolitan University. The assessment task was a set of five multiple choice questions on the topic of constructors in the context of object-oriented programming. A paper-based assessment session was undertaken by students who were registered for the module and 115 student responses were captured. The multiple choice questions are attached in Appendix B.

In both assessment tasks, the session was carried out in an environment where students were not allowed to copy each other and get help from others. Captured student responses have to be processed and saved as text file with predefined format where each student attempt is represented by a sequence of characters. The number of characters corresponds to the number of multiple choice questions. The distribution of the student responses captured from the first and second assessment tasks are shown in figure 3.2 and 3.3 respectively.

A sequence of characters that represent a student attempt is transformed into a binary vector using a coding of information well known in the field of infor-

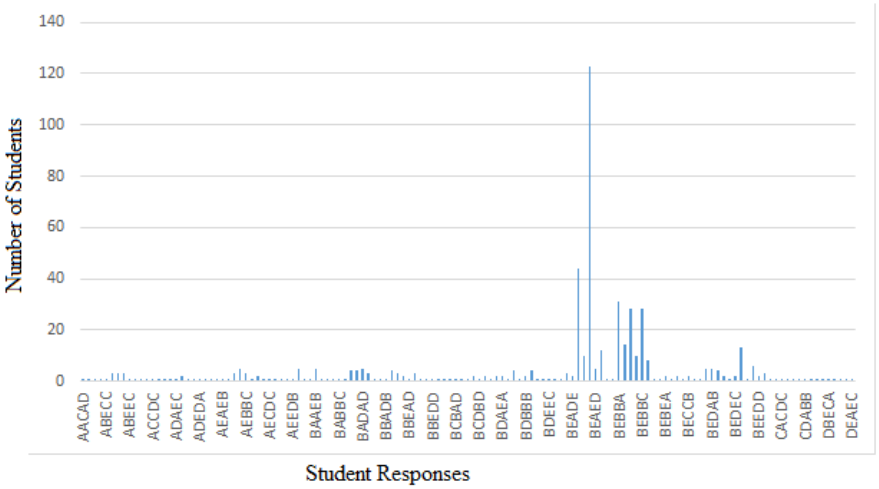


Figure 3.2: A bar chart of the distribution of student responses captured from assessment task one

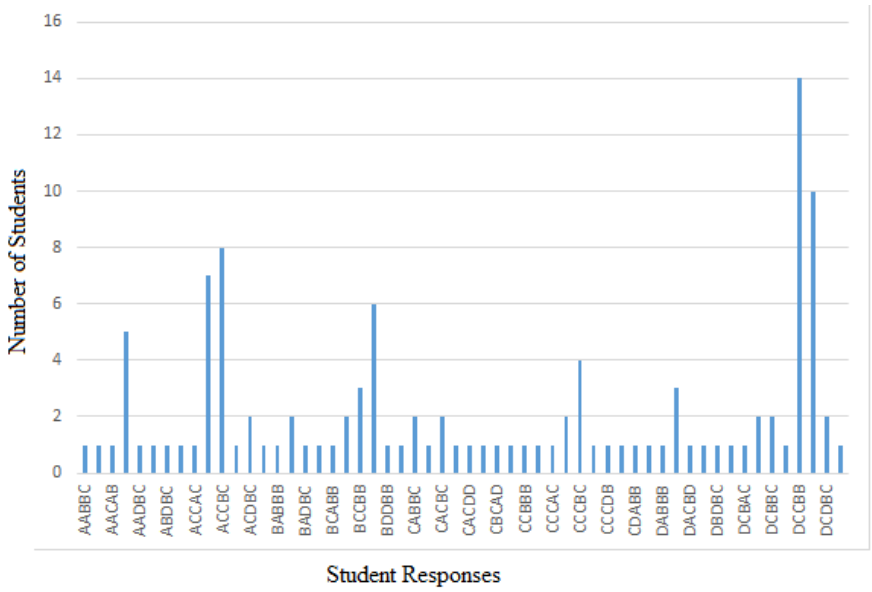


Figure 3.3: A bar chart of the distribution of student responses captured from assessment task two

mation theory. Based on this coding scheme, each student answer (character) is represented by a binary number with a number of digits equal to the number of options of a multiple choice question. The position of 1 is changed to represent all possible answers. For example, to represent a sequence of answers with four options, we use four digits binary numbers: $0001, 0010, 0100, 1000$ to represent A, B, C, D respectively. According to this representation scheme, the representation of three student responses to a set of three MCQs with four options are shown as follows:

Table 3.1: A coding example for three responses to a set of three MCQs with four options

Response	**Binary code**
AAB	000100010010
AAC	000100010100
AAD	000100011000

The euclidean distance between response one and two is 1.41 and is the same as the euclidean distance between response one and three. On the other hand, the hamming distance between response one and two is 2 which is also the same as the hamming distance between response one and three. As a result, the representation scheme managed to retain the similarity information between the three responses when they are mapped to binary vectors, which makes it appropriate to represent a sequence of characters as a binary vector.

3.4 Student Group Profiles

The output of the Snap-drift learning agent consists of a group allocation map, a set of winning nodes of s layer and two matrices of weight vectors corresponding to the distributed and selection modules. The set of winning nodes represent the different student groups while the two matrices of weight vectors can be used to identify the appropriate student group of a new student response. The group allocation map contains information about the allocation of each training pattern to its appropriate student group.

As mentioned previously, the purpose of identifying student groups is to help tutors in revealing gaps of understanding and misconceptions so that they can write an appropriate diagnostic feedback that improves student learning performance. Therefore, it is necessary to represent the group allocation map in a format that facilitates achieving this purpose. The proposed method to construct student group profile based on the group allocation map is described using a pseudo-code as follows.

1. Set group profile threshold that determines the most likely answer
2. Get group allocation map from the Snap-Drift learning agent.
3. Create an empty matrix for each student group.
4. Set the row and column of each matrix to the number of multiple choice questions and the number of options respectively.
5. Compute the value of each element (x_{ij}) of all matrices as the percentage of student responses for question i who answered option j based on the group allocation map.
6. FOR each matrix
    (a) FOR each row
        i. Find the column index with the highest percentage.

ii. IF the highest percentage is greater than the threshold THEN
Replace the row with an appropriate character ($1 = A, 2 = B, 3 = C, 4 = D, 5 = E$) corresponding to the column index

iii. ELSE
Replace the row with an asterisk (*).

iv. ENDIF

(b) ENDFOR

7. ENDFOR

8. Return matrices

The implemented Snap-Drift modal learning neural network could output different set of student group profiles based on the values of the learning parameters. To get insight into the relationship among the learning parameters and how they affect the output, several combinations of the learning parameters were tested using iris data set and the training patterns prepared in the previous section. A graphical user interface was integrated with the implemented snap-drift modal learning neural network to assist the testing. As shown in figure 3.4, the graphical user interface comprises four components: training data panel, SDNN parameters panel, control granularity of student groups panel, training panel and output visualisation panel. The output visualisation panel component is integrated with MATLAB.

Based on the observation of the tests, the number of nodes in d layer combined with its number of simultaneously active nodes (D) and quality assurance threshold (hurdle) determine the number and nature of groups. Generally, increasing the number of features (D) in d layer decreases the number of groups. For a given number of features, increasing the quality assurance threshold tends to increase the number of groups. The quality assurance threshold can be used

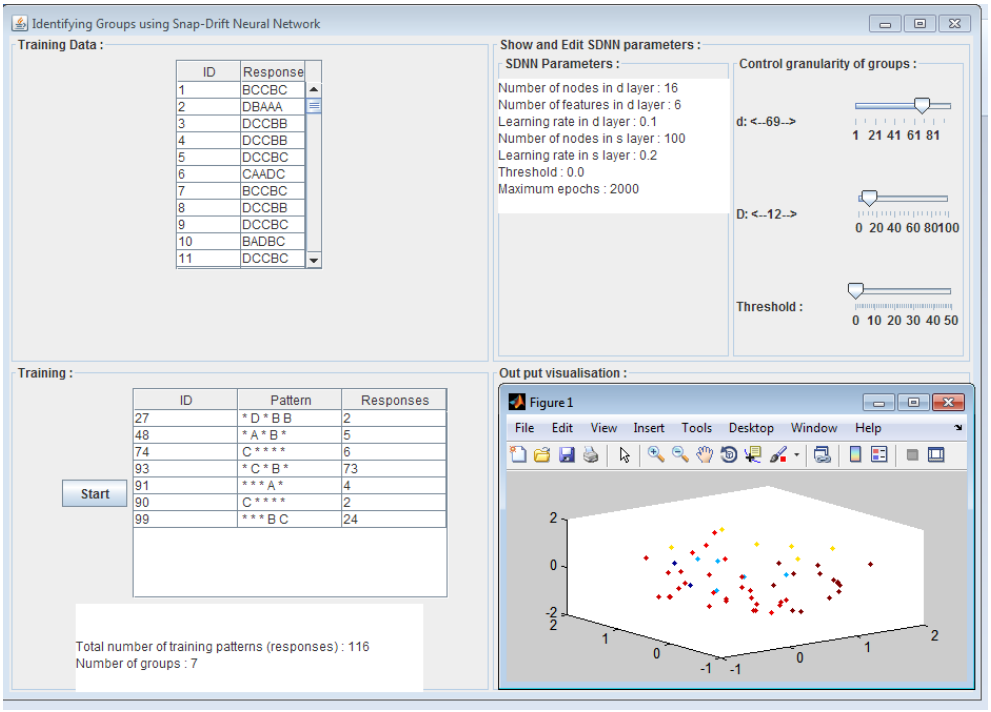


Figure 3.4: Screen shot of Graphical user interface of SDNN tool

to fine tune the nature of the identified groups. The influence of the threshold starts after a certain value which is correlated to the number of features in d layer. When its value is greater than a certain value, it becomes hard to identify groups due to slow convergence speed and lack of available uncommitted nodes in s layer. The learning rates for d and s layers influence the convergence speed. The recommended values of the learning rates for d and s layers are 0.1 and 0.2 respectively. Convergence is not guaranteed. If there is no convergence, the trivial solution is to re-start the training process or increase the quality assurance threshold by a small amount.

Generally, too many student groups require too much tutors' time in writing feedback whereas a very small number of student groups could be less effective as the feedback tends to be too generic. Therefore, it is very important to de-

termine an appropriate number of student groups in order to construct effective diagnostic feedback. As long as the training data set is large and representative of the potential patterns, the implemented SDNN is capable of converging to an appropriate number of student groups. However, the implemented SDNN might find a small or large number of student groups because of less representative and inadequate training patterns. In this case, we can adjust the value of D, d and quality assurance threshold to decrease or increase the identified student groups.

Based on the method described in the above, a student group is represented by a sequence of most likely responses for each multiple choice question. The method specifies a threshold for determining the most likely responses. Its value should be between 70 and 80. A value below 70 decreases the number of student responses assigned to their appropriate student group, whereas a value above 80 decreases the probability of finding most likely sequence of responses. Even though the group profile threshold is set between 70 and 80, there might be a mix of responses to a particular question. This case happens when percentages of student responses for a given question is evenly distributed over all possible options.

If a student group is represented by a sequence of asterisk (*), which means a mix of responses for all multiple choice questions, it is not a useful student group since it doesn't reveal any gaps of understanding and misconceptions of a particular topic. The question is what should be the nature or pattern of the student group profile in order to be useful. Since one multiple choice question can't assess any concept or aspect of a given topic, a student group profile of only one most likely response can't be used in revealing any gaps of understanding and misconceptions of a given topic. Therefore, a student group profile should

be represented by at least two most likely responses in order to be useful. That is why it is important to define the characteristics of each student group and the number of student groups as criteria for assessing the usefulness of a set of student group profiles for effective diagnostic feedback.

Once the criteria for assessing the usefulness of a set of student group profiles was defined, the implemented SDNN integrated with a graphical user interface was applied to the two training data sets prepared in the previous section. Using the defined criteria and the insight gained into the relationship among the learning parameters, two sets of student group profiles were developed corresponding to the two assessment tasks described in the previous section. The two sets of student group profiles are shown in figure 3.5 and figure 3.6. The learning parameters used were 2000 maximum epochs and 0.1 and 0.2 learning rates and 100 number of s nodes, which is large enough for allocating uncommitted nodes and identify up to 100 student groups.

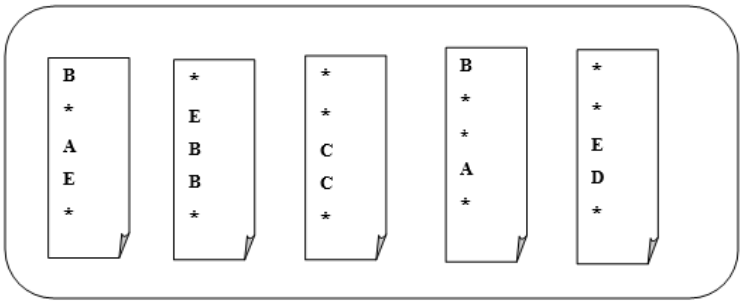


Figure 3.5: Student group profiles for the first assessment task

If the learning parameters are changed, the identified student group profiles will change slightly. The change might result in more student group profiles or less student group profiles. The increase in the number of student group profiles is because of the break-up of a student group profile into two or more. The

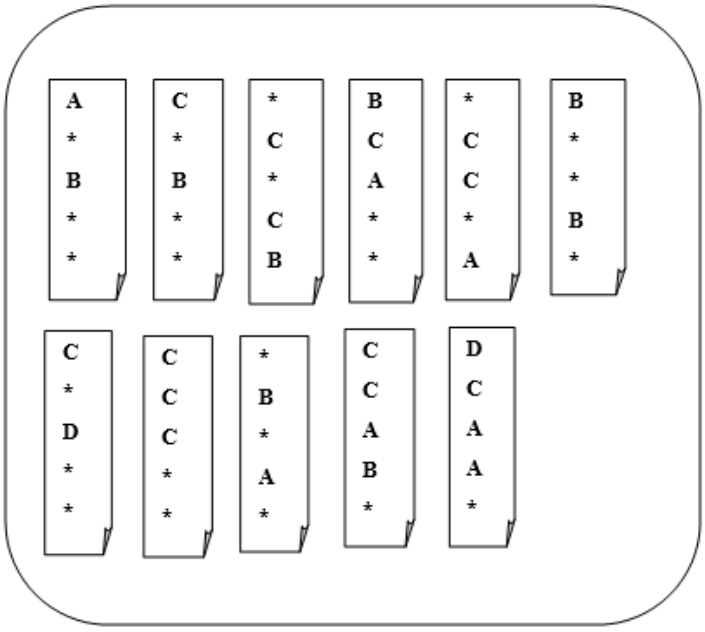


Figure 3.6: Student group profiles for the second assessment task

decrease in the number of student group profiles is due to the aggregation of two or more student group profiles into one. The criteria for assessing usefulness of a set of student group profiles will guide a tutor in identifying a set of student group profiles which dont consume much time in writing diagnostic feedback and which provide specific diagnostic feedback.

3.5 Diagnostic Feedback Construction

Feedback is the most essential part of formative assessments as the sole purpose of performing formative assessment is to provide feedback based on students' responses observed from assessments [7]. Feedback can be defined as an information communicated to a learner that is intended to modify the learner's thinking or behaviour for the purpose of improving learning [58]. A formative assessment

without feedback has no effect on improving student learning experience and a formative assessment with feedback does not necessarily improve student learning performance [10]. For example, if answers are included as part of feedback, the effect of feedback on student learning will be negative [10; 13]. The quality of feedback determines the effectiveness of formative assessment [10].

Several research studies have been conducted to find out what characteristics of feedback actually improve student learning. Most researchers agree that for feedback to be effective, it should be non-evaluative, supportive, timely and specific and should include the comparison of actual performance with some established standard of performance [58]. Researchers have also reported that the content of effective feedback should contain both verification and elaboration [58]. Verification indicates whether student's work is correct or not. Elaboration is an information that provides details of how to improve an answer [58]. The Elaboration aspect of feedback can be more specific and directive when it addresses a topic, a response or a particular error. It can also be more general and facilitative when it provides worked examples or gives gentle guidance.

A diagnostic feedback is generated based on the analysis of student responses that addresses gaps of understanding and misconceptions[4; 53]. It should confirm the gaps of understanding or misconceptions and provide details on how to close the gaps of understanding and correct misconceptions without specifying which questions are correct or not. To support construction of diagnostic feedback, student responses are represented by student group profiles as described in the previous section. Instead of writing for each student response, a diagnostic feedback is constructed per student group profile.

A combination of student responses, which can be extracted from a set of

student group profiles, reveals which concepts are already understood and/or which are misunderstood. This information helps tutors in writing the verification part of a diagnostic feedback. Learning outcomes of a particular topic, which specify what students should be able to do at the end of a teaching session, define a set of related concepts for the topic and their level of understanding. Tutors can use this information to write the elaboration part of a diagnostic feedback that details how students close gaps of understanding and correct misconceptions.

If you consider the first student group profile from the set of student group profiles for the first assessment task shown in figure 3.5, most students respond option B for question 1, option A for question 3 and option E for question 4. By looking at this combination of responses, learning outcomes of the assessment task and the multiple choice questions, we can understand that the students have difficulty in rounding calculated probability values and choosing a right event type, however, they have understood the concept of probability and conditional probability. The second and fifth asterisk (*) indicate that there are no most likely responses for question 2 and 5 respectively. Generic feedback that addresses all incorrect options of both questions can be included. The diagnostic feedback for the remaining student group profiles were constructed based on the same procedure described for the first student group profile. The diagnostic feedback constructed for both assessment tasks are shown below.

Table 3.2 shows the diagnostic feedback for the first student group of the first assessment group. It is constructed based on the student group profile in figure 3.5.

Table 3.2: Diagnostic feedback for the first student group of the first assessment task

You have understood the concept of probability and conditional probability. However, pay attention to appropriate rounding of calculated probability values and also make sure that the right event is considered. For instance, adult does not mean only male or female, it includes both male and female. If the probability that A occurs given the event B has occurred ($P(A/B)$) is equal to the probability that A occurs ($P(A)$), then A and B are said to be independent or unrelated. For example, if we throw a coin and get a head, this outcome will not affect the outcome of a second throw, as a result these events are independent.

Table 3.3 shows the diagnostic feedback for the second student group of the first assessment group. It is constructed based on the student group profile in figure 3.5.

Table 3.3: Diagnostic feedback for the second student group of the first assessment task

You have understood the concept of basic probability, you need only to make sure that appropriate rounding of calculated probability values is chosen. The concept of conditional probability is not understood well. Conditional probability is not calculated the same way as probability without condition. The probability that A occurs given the event B has occurred ($P(A/B)$) is equal to the probability that both A and B ($P(AandB)$) divided by probability of B ($P(B)$). So, when conditional probabilities are calculated use the total of the condition (total of event B) in the denominator rather than the overall total. If the probability that A occurs given the event B has occurred ($P(A/B)$) is equal to the probability that A occurs ($P(A)$), then event A and B are said to be independent or unrelated. For example, if we throw a coin and get a head, this outcome will not affect the outcome of a second throw, as a result these events are independent.

Table 3.4 shows the diagnostic feedback for the third student group of the first assessment group. It is constructed based on the student group profile in

Table 3.4: Diagnostic feedback for the third student group of the first assessment task

Probability of A is calculated by dividing the number of A events by the total number of events. When probabilities are calculated make sure that appropriate rounding of calculated values is chosen. The conditional probability is not understood well. Conditional probability is not calculated the same way as probability without condition. The probability that A occurs given the event B has occurred ($P(A/B)$) is not equal to the probability that B occurs given the event A has occurred ($P(B/A)$). Pay attention to the question to identify the event condition. If the probability that A occurs given the event B has occurred ($P(A/B)$) is equal to the probability that A occurs ($P(A)$), then event A and B are said to be independent or unrelated. For example, if we throw a coin and get a head, this outcome will not affect the outcome of a second throw, as a result these events are independent.

figure 3.5.

Table 3.5: Diagnostic feedback for the fourth student group of the first assessment task

You have understood the concept of basic probability, however, the conditional probability is not understood well. Conditional probability is not calculated the same way as probability without condition. The probability that A occurs given the event B has occurred ($P(A/B)$) is not equal to the probability that B occurs given the event A has occurred ($P(B/A)$). Pay attention to the question to identify the event condition. If the probability that A occurs given the event B has occurred ($P(A/B)$) is equal to the probability that A occurs ($P(A)$), then event A and B are said to be independent or unrelated. For example, if we throw a coin and get a head, this outcome will not affect the outcome of a second throw, as a result these events are independent.

Table 3.5 shows the diagnostic feedback for the fourth student group of the first assessment group. It is constructed based on the student group profile in figure 3.5.

Table 3.6: Diagnostic feedback for the fifth student group of the first assessment task

You have understood the conditional probabilities very well. However, you make some errors regarding the basic probability. Probability of A is calculated by dividing the number of A events by the total number of events. When probabilities are calculated make sure that appropriate rounding of calculated values is chosen and the overall total is calculated correctly. If the probability that A occurs given the event B has occurred ($P(A/B)$) is equal to the probability that A occurs ($P(A)$), then event A and B are said to be independent or unrelated. For example, if we throw a coin and get a head, this outcome will not affect the outcome of a second throw, as a result these events are independent.

Table 3.6 shows the diagnostic feedback for the fifth student group of the first assessment group. It is constructed based on the student group profile in figure 3.5.

Table 3.7: Diagnostic feedback for the first student group of the second assessment task

A constructor is a special method that is called when you create class instances (objects). It is not like any other instance methods since it does not return or change value of fields. Constructors are used to initialise the fields of objects to user input values. They can also set the value of a field which is not specified as parameter to its default value. When you create objects using a constructor, the default values are not supplied as parameters. The header of constructors is similar to method headers, however, the name of a constructor is the same as its class name and no return type is specified explicitly.

Table 3.7 shows the diagnostic feedback for the first student group of the second assessment group. It is constructed based on the student group profile in figure 3.6.

Table 3.8: Diagnostic feedback for the second student group of the second assessment task

A constructor is a special method that is called when you create objects. It is not like any other instance methods since it does not return or change value of fields. Once you create an object using a constructor, you can store it in a variable that refers objects. Before you use any variable it has to be declared first. For example, to declare a variable named temp that refers objects of type String, you write the following statement: public String temp; Constructors are used to initialise the fields of objects to user input values. They can also set the value of a field which is not specified as parameter to its default value. When you create objects using a constructor, the default values are not supplied as parameters. The header of constructors is similar to method headers, however, the name of a constructor is the same as its class name and no return type is specified explicitly.

Table 3.8 shows the diagnostic feedback for the second student group of the second assessment group. It is constructed based on the student group profile in figure 3.6.

Table 3.9: Diagnostic feedback for the third student group of the second assessment task

A constructor is a special method that is called when you create class instances (objects). It is not like any other instance methods since it does not return or change value of fields. Constructors are used to initialise the fields of objects to user input values. They can also set the value of a field which is not specified as parameter to its default value. When you create objects using a constructor, the default values are not supplied as parameters. The header of constructors is similar to method headers, however, the name of a constructor is the same as its class name and no return type is specified explicitly. The header includes access modifier (public), name of the constructor and optional list of parameters. The constructor header is different from class declaration, which includes access modifier (public), class key word and name of class. For example to declare a class called Square, you write "public class Square"

Table 3.9 shows the diagnostic feedback for the third student group of the second assessment group. It is constructed based on the student group profile in figure 3.6.

Table 3.10: Diagnostic feedback for the fourth student group of the second assessment task

You have understood the concept of constructors. However, you need to know how to write the header of a constructor and how to call a constructor to create objects. Constructors are used to initialise the fields of objects to user input values. They can also set the value of a field which is not specified as parameter to its default value. When you create objects using a constructor, the default values are not supplied as parameters. The header of constructors is similar to method headers, however, the name of a constructor is the same as its class name and no return type is specified explicitly. The header includes access modifier (public), name of the constructor and optional list of parameters.

Table 3.10 shows the diagnostic feedback for the fourth student group of the second assessment group. It is constructed based on the student group profile in figure 3.6.

Table 3.11: Diagnostic feedback for the fifth student group of the second assessment task

A constructor is a special method to create class instances (objects). It is not like any other instance methods since it does not return or change value of fields. Constructors are used to initialise the fields of objects to user input values. They can also set the value of a field which is not specified as parameter to its default value. When you create objects using constructors, the default values are not supplied as parameters. The header for constructors is similar to method headers, however, the name of a constructor is the same as its class name. The header includes access modifier (public), name of the constructor and optional list of parameters.

Table 3.11 shows the diagnostic feedback for the fifth student group of the second assessment group. It is constructed based on the student group profile in figure 3.6.

Table 3.12: Diagnostic feedback for the sixth student group of the second assessment task

A constructor is a special method to create class instances (objects). It is not like any other instance methods since it does not return or change value of fields. No return type is specified explicitly for a constructor. Constructors are used to initialise the fields of objects to user input values. They can also set the value of a field which is not specified as parameter to its default value. When you create objects using a constructor, the default values are not supplied as parameters.

Table 3.12 shows the diagnostic feedback for the sixth student group of the

second assessment group. It is constructed based on the student group profile in figure 3.6.

Table 3.13: Diagnostic feedback for the seventh student group of the second assessment task

A constructor is a special method to create class instances (objects). It is not like any other instance methods since it does not return or change value of fields. Constructors are used to initialise the fields of objects to user input values. They can also set the value of a field which is not specified as parameter to its default value. When you create objects using a constructor, the default values are not supplied as parameters. The header of constructors is similar to method headers, however, the name of a constructor is the same as its class name and no return type is specified explicitly. Even if constructors do not return values, there is no need to add void before a constructor name.

Table 3.13 shows the diagnostic feedback for the seventh student group of the second assessment group. It is constructed based on the student group profile in figure 3.6.

Table 3.14: Diagnostic feedback for the eighth student group of the second assessment task

A constructor is a special method to create objects. It is not like any other instance methods since it does not return or change value of fields. Once you create an object using a constructor, you can store it in a variable that refers objects. Before you use any variable it has to be declared first. For example, to declare a variable named temp that refers objects of type String, you write the following statement: public String temp; Constructors are used to initialise the fields of objects to user input values. They can also set the value of a field which is not specified as parameter to its default value. When you create objects using a constructor, the default values are not supplied as parameters.

Table 3.14 shows the diagnostic feedback for the eighth student group of the second assessment group. It is constructed based on the student group profile in figure 3.6.

Table 3.15: Diagnostic feedback for the ninth student group of the second assessment task

A constructor is a special method to create class instances (objects). It is not like any other instance methods since it does not return or change value of fields. Constructors are used to initialise the fields of objects to user input values. They can also set the value of a field which is not specified as parameter to its default value. When you create objects using a constructor, the default values are not supplied as parameters. The header of constructors is similar to method headers, however, the name of a constructor is the same as its class name and no return type is specified explicitly. Even if constructors do not return values, there is no need to add void before a constructor name.

Table 3.15 shows the diagnostic feedback for the ninth student group of the second assessment group. It is constructed based on the student group profile in figure 3.6.

Table 3.16: Diagnostic feedback for the tenth student group of the second assessment task

You have understood the concept of constructors, but you need to understand the difference between creating objects and declaring variables that refer objects. Once you create an object using a constructor, you can store it in a variable that refers objects. Before you use any variable it has to be declared first. For example, to declare a variable named temp that refers objects of type String, you write the following statement: public String temp; Constructors are used to initialise the fields of objects to user input values. They can also set the value of a field which is not specified as parameter to its default value. When you create objects using a constructor, the default values are not supplied as parameters.

Table 3.16 shows the diagnostic feedback for the tenth student group of the second assessment group. It is constructed based on the student group profile in figure 3.6.

Table 3.17: Diagnostic feedback for the eleventh student group of the second assessment task

A constructor is a special method to create class instances (objects). It is not like any other instance methods since it does not return or change value of fields. Constructors are used to initialise the fields of objects to user input values. They can also set the value of a field which is not specified as parameter to its default value. When you create objects using a constructor, the default values are not supplied as parameters. The header of constructors is similar to method headers, however, the name of a constructor is the same as its class name and no return type is specified explicitly. Even if constructors do not return values, there is no need to add void before a constructor name.

Table 3.17 shows the diagnostic feedback for the eleventh student group of the second assessment group. It is constructed based on the student group profile

in figure 3.6.

3.6 Assessment Sessions

A web-based formative assessment tool was designed and implemented to conduct assessment sessions so that students are able to undertake assessment tasks based on multiple choice questions, receive a diagnostic feedback based on their responses instantly, attempt multiple times and record information about the session. The tool integrates five components, which are the implemented snap-drift modal learning neural network, assessment tasks based on multiple choice questions, diagnostic feedback, assessment session manager and a relational database. The tool was implemented using web technologies (XHTML, CSS, Java script and AJAX), Java technologies (JSP and Servlet), MySql relational database server and Hibernate for mapping relational tables to Java objects.

The implemented snap-drift modal learning neural network first identifies groups which are then represented as student group profiles during training phase and it is able also to assign a student group profile for a new student response. The relational database component stores assessment tasks, constructed diagnostic feedback and information about assessment session such as student responses, attempts, duration between attempts, time and date of sessions. The assessment session manager manages the whole assessment process, which includes displaying an assessment task, capturing student responses, getting the appropriate diagnostic feedback and delivering it to a student, and recording all relevant information about a session.

Table 3.18: A sample of data recorded from the first trial

Student Number	**Response**	**Attempt Number**	**Duration in Min**	**Winning Node**	**Group Number**
1	BEAEA	1	16.6	3	1
	BEAEA	2	1.4	3	1
	BEEDA	3	1.86	23	5
	BEAEA	4	1.17	3	1
	BEADA	5	0.76	3	1
2	CCECC	1	15.6	11	3
	CBAEB	2	1.3	3	1
3	BEBBC	1	8.6	9	2
4	BADAB	1	7.7	12	4
	BAAEB	2	16.0	3	1
5	BBEDA	1	4.6	23	5
6	BEBBB	1	11.0	9	2
7	BEBBA	1	4.7	9	2
	BEAEA	2	16.6	3	1
	BEAEA	3	16.6	3	1
8	BEAEA	1	16.6	3	1
9	BEAEA	1	16.6	3	1
	BEAEA	2	16.6	3	1
	BEAEA	3	16.6	3	1
	BEAEA	4	16.6	3	1

Two trials were conducted using the developed web-based formative assessment tool during workshops. During trials, students were not allowed to copy from each other. A different cohort of students, who didn't participate during a training phase, were chosen to participate in the trials. All selected students were registered for Introduction to Data Analysis module for the first trial and Introduction to Programming module for the second trial. Before the trials, they were exposed to the selected topics of the modules. Forty nine students participated during the first trial and twenty five students during the second trial. Samples of the gathered data from the two trials are shown in tables 3.18 and 3.19.

Table 3.19: A sample of data recorded from the second trial

Student Number	**Response**	**Attempt Number**	**Duration in Min**	**Winning Node**	**Group Number**
1	ACBCB	1	2.0	2	3
	BADBA	2	3.4	5	6
	CABDA	3	1.0	1	2
2	BCBCD	1	4.0	3	4
	BCABC	2	3.4	3	12
3	BCAAA	1	1.9	3	4
	BCABD	2	1.4	3	4
	BCABB	3	0.3	3	4
	BCABC	4	0.9	3	12
4	CCADA	1	2.7	9	10
	CCACB	2	1.3	9	10
5	BCABB	1	2.8	3	4
	BCABC	2	0.1	3	12
6	ACAAB	1	3.0	10	11
7	CCABB	1	4.7	9	10
	CCABA	2	1.9	9	10
	BCABA	3	1.1	3	4
	BCABC	4	1.2	3	12
8	CAADA	1	1.0	9	10
	ABBAC	2	1.0	0	1
	DCBBB	3	0.5	2	3
	BCCAC	4	0.2	7	8
	BCCCC	5	0.2	3	4
	BCCCB	6	0.4	3	4
9	CCABA	1	3.2	9	10
	CCABB	2	0.8	9	10
	BCABC	3	0.4	3	12

3.7 Results and Discussion

To assess whether each student response is assigned to an appropriate student group profile, all student responses and their corresponding student group profile were extracted from the database recorded using the web-based formative assessment tool during the two trials. The extracted student responses are shown in figure 3.7 and figure 3.8 from the first and second trials respectively.

Fifteen unique student responses were assigned to student group profile one from the first trial. By comparing the student group profile of student group one with the assigned student responses, six responses matched perfectly with the group profile pattern. For the remaining responses, there are only two matches out of the three for seven responses and one match for two responses.

The remaining student group profiles were compared from both trials with their respective assigned student responses and the student responses that are partially matched were examined if they could be assigned to any other existing student group profiles. The result showed that none of them could be assigned to any other student group profiles. This implies that the trained snap-drift neural network has managed to assign those partially matched responses to the closest student group as much as possible. The percentage of perfectly matched student responses could be improved by increasing the number of student group profiles and selecting student group profiles with no more than three most likely responses.

To assess the effectiveness of the diagnostic feedback in improving learning performance of students, data extracted from the database recorded using the web-based formative assessment were analysed. The data set includes the se-

Group Number	Group Profile	Student Responses	Group Number	Group Profile	Student Responses
1	B * A E *	BEAEA	2	* E B B *	BEBBC
		BEADA			BEBBB
		CBAEB			BEBBA
		BAAEB			AEBBD
		BEAED			BEBBD
		BEBEC			AEBBB
		BEAEB			BBBBB
		CBCEA			BDBBA
		AEAEB			AEBBB
		BBAED			AEBBA
		BEBEB			BEEBA
		BEAEC			AEBBC
		BEBEA			
		CADEB	4	B * * A *	BADAB
		BEADB			BEDAD
					BCBAC
3	* * C C *	CCECC			BEDAA
		BCECD			BAAAA
		ABCCA			BADAC
		ACCCC			BADAB
					BADAA
					BEDAD
					BADAA
5	* * E D *	BEEDA			
		BBEDA			
		BEEDD			
		BECDA			
		ADEDC			
		AEADA			
		BACDC			
		BEEDC			
		BEBDA			
		BEEDB			
		BAEDD			
		DCBDA			

Figure 3.7: Unique student responses extracted and sorted according to their corresponding student group profile for the first trial

Group Number	Group Profile	Student Responses	Group Number	Group Profile	Student Responses	Group Number	Group Profile	Student Responses
1	A * B * *	ABBAC	2	C * B * *	CABDA	3	* C * C B	ACBCB
		ABBCB			BBBDA			DCBBB
		ABBBA			CCBCA			BCCCB
		AABCA						DDDDB
4	B C A * *	BCAAD	5	* C C * A	BCCCA	6	B * * B *	BADBA
		BCABB			ACCCA			BBBBC
		BCABD						BCDBB
		BCABA						BCCBB
		ACACC						BCBBB
		BCADA						BCDBC
		BCACA						
		BCABC	7	C * D * *	No responses	8	C C C * *	BCCAC
		BCAAA						
		BCBCD						
		BCDCC						
		BCAAB						
		BBACB	9	* B * A *	BBBAB	10	C C A B *	CCADA
		BCACB			BBAAA			CCACB
		BCDDA			CBCAB			CCABB
		BCCDD			CBAAB			CCABA
		BCACD			CBDBA			CAADA
					CBCAA			CAABB
								CCAAA
11	D C A A *	ACAAB						

Figure 3.8: Unique student responses extracted and sorted according to their corresponding student group profile for the second trial

quence of attempts per each student, the assigned student group profile and their corresponding diagnostic feedback for each attempt and duration between consecutive attempts. A state transition diagram was used to visualise the interaction among the student responses, student group profiles and diagnostic feedback. Figure 3.9 and 3.10 show the possible transition of students for the first trial and second trial respectively. The different student group profiles are identified as states. The state of a student is determined by his/her current responses and a diagnostic feedback is considered as an event that can trigger transition from one student group profile to another. The transition of states is represented by an arrow. A student group profile is characterised by a group number and average score. The starting state of a student is determined by the first response and can be at the final state if he/she answers all questions correctly.

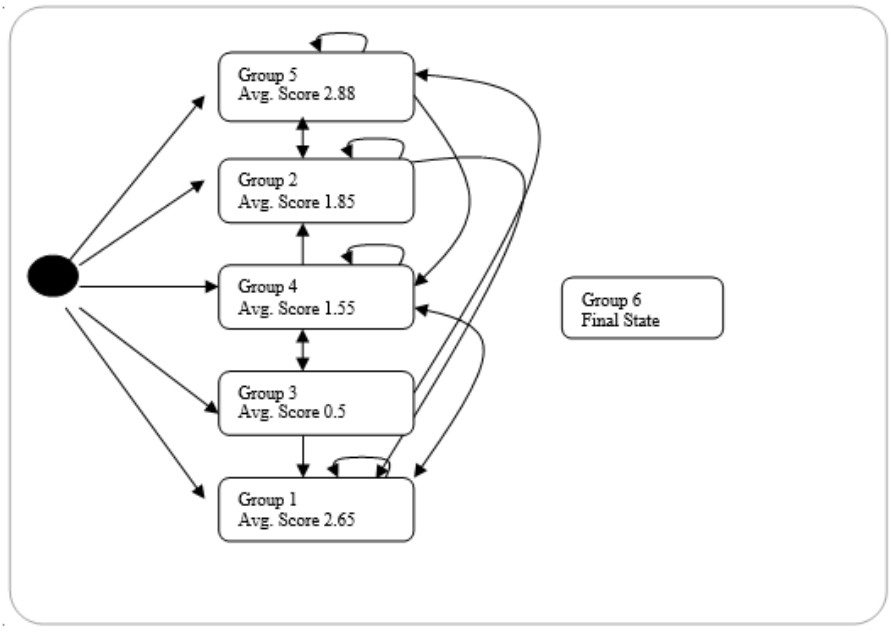


Figure 3.9: Visualising the possible transition of students from one student group profile to another using transition state diagram for the first trial

From the above state transition diagram shown in figure 3.9, we can understand that all student group profiles can be a starting point and students were changing states within the five intermediate states. However, there is no link between the intermediate states and the final state and there is no clear pattern that shows a learning path to reach the final state. Possible reasons for this might be difficulty of the questions or little effort in reading diagnostic feedback as it was revealed from the information about the duration between consecutive attempts.

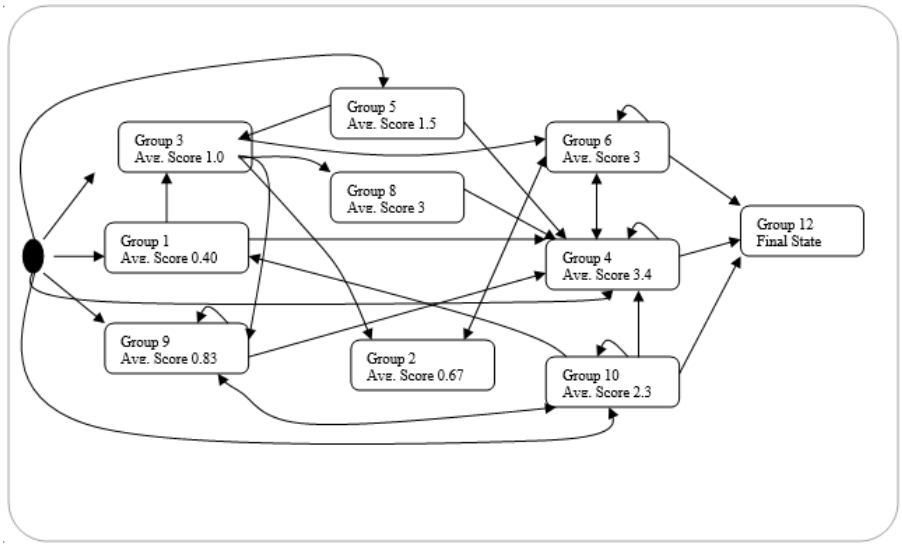


Figure 3.10: Visualising the possible transition of students from one student group profile to another using transition state diagram for the second trial

There are different starting points in the above state transition diagram shown in figure 3.10, which demonstrates that students were at different group states with different gaps of understanding and misconceptions when they start the assessment session. During the assessment session, some students managed to reach the final state and they followed a pattern to reach the final state. The state transition diagram revealed that students have to reach group states 6, 4, and 10

before they move to the final state. Most students start from low performance (low average score) group states and move toward the high performance group states and finally reach the final state. This indicates that the diagnostic feedback has a positive impact on improving the learning performance of students.

If the assessment is repeated, the student grouping does not change. The student grouping was determined during the training phase. However, the state transition diagram might change since it depends on the current level of students' understanding and how they react to the diagnostic feedback. When the same students repeat the assessment, we will see a different state transition diagram since most students will improve their learning experience from the first assessment

3.8 Summary

Based on previous research studies that investigate the application of snap-drift modal learning neural networks in analysing student responses to a set of multiple choice questions, an interactive software tool was developed. The tool implements a snap-drift modal learning neural network integrated with visualisation feature of MATLAB. The implemented snap-drift modal learning neural network was tested using iris data set. The result demonstrated that it was implemented correctly and was capable of classifying linearly separable data. The average performance for classifying non-linearly separable classes, which are the second and third classes, was above 95 percent. The developed software tool is capable of analysing the behaviour of snap-drift learning algorithm and the effect of learning parameters on the identified student groups.

The proposed method details an algorithm for profiling outputs of snap-drift modal learning neural networks, a criteria for assessing usefulness of student group profiles in revealing gaps of understanding and misconceptions of a particular topic, and a guideline for supporting tutors in writing diagnostic feedback based on profiled student groups.

The method was applied to two real assessment tasks designed for two topics selected from Introduction to Data Analysis and Introduction to Programming modules in order to gather student responses and identify two sets of student group profiles. Two trials were also conducted using a developed web-based formative assessment tool during workshops to evaluate the effectiveness of the proposed method. Forty nine students participated during the first trial and twenty five students during the second trial. Analysis of gathered student responses showed that all of them were assigned to their appropriate student group profiles and the percentage of perfectly matched student responses could be improved by increasing the number of student group profiles and selecting student group profiles with no more than three most likely responses. The analysis of gathered student responses also showed that the diagnostic feedback constructed based on the identified student group profiles has a positive impact on improving the learning performance of students.

The insight gained into the application of snap-drift modal learning neural network for modelling student responses based on multiple choice questions will be applied for extending its application in modelling student responses gathered from programming exercises and development of an intelligent web-based formative assessment for facilitating conceptual understanding of topics.

Chapter 4

Unsupervised Modelling of Object-oriented Programming Exercise Responses

As described in chapter three, a snap-drift modal learning neural network was applied successfully in identifying student group profiles based on responses gathered from multiple choice questions. The student group profiles represent different categories of understandings of a given topic or sub topic. The main research question addressed in this chapter is how to extend the application of snap-drift to model student responses gathered from programming exercises.

Even though multiple choice questions could be used in principle to assess students ability to recall knowledge, understand concepts, and apply knowledge and understanding in order to solve real life problems, it is not effective to assess programming ability of students. A free response assessment method such as programming exercise is necessary for students to write programs to demonstrate

their understanding of programming concepts and programming abilities. The purpose of modelling student responses to programming exercises is for identifying student group profiles that facilitate generation of diagnostic feedback to improve development of basic object-oriented programming abilities for novice programmers. The focus is on designing and coding object-oriented programming abilities. Requirement analysis and testing programming abilities are not considered as the emphasis is to develop basic object-oriented programming abilities for novice programmers.

In this chapter object-oriented programming approach is introduced first. Secondly, a method for extracting features to represent student responses to programming exercises is described. Thirdly, a method for parsing the defined features based on the defined features is described. Fourthly, how a snap-drift modal learning neural network can be applied to identify student group profiles that represent different levels of basic object-oriented programming abilities, is explained. Fifth, identifying and developing student group profiles and then generating diagnostic feedback are described. Finally, a summary is presented.

4.1 Object-oriented Programming Approach

Programming is the most important skill to acquire for computer science students. That is why in most higher education programming modules are compulsory for any first year student enrolled to study Computer Science. Before adopting object-oriented programming approach to teach first year studying Computer Science, most universities were teaching procedural languages such as Pascal, Basic, Fortran and C. Procedural languages are languages in which the basic

unit of abstraction is a procedure or an algorithm and there is a clear distinction between an algorithm and the data it operates on [22]. Object-orientation is an approach to systems development where computer programs are organised using objects. The fundamental difference between this approach and the procedural approach is that data is encapsulated with the functions that act upon it [9].

The teaching approach for procedural languages is a bottom-up approach, which starts with the basics or fundamentals of programming and then slowly builds knowledge and programming skill over time. In this approach, students are taught first the concept of variables, how control structures such as if and loop statements work, the details of parameter passing and finally how to implement algorithms and data structures to solve programming problems [22]. The problem with this bottom-up approach is that the main issues of problem solving, design and software engineering are overlooked [22]. That is why object-oriented programming was adopted since it supports top-down teaching approach. The other obvious reason was because of the popularity and application of object-oriented programming languages such as Java and C++ in several industries.

In top-down teaching approach, which is also known as object first, initially students use the object world by interacting with sample programs that are provided to illustrate specific object-oriented programming features [22; 39]. Once students are familiar with objects, they can start learning how an object is implemented using a specific programming language such as Java or C++ and then, they can be taught how to modify and extend an existing implemented object to suit the needs of a particular program. Finally, they are taught how to define their own objects and implement them. The details of the programming world, in particular, the algorithmic details of object-oriented programming languages

are revealed slowly and only as needed to illustrate the principles of objects [22]. In addition to this, the algorithmic constructs such as assignment, loops, conditional statements should be introduced as they are important to manipulate and support objects [22].

The most important object-oriented concepts are class, instance, generalisation and specialisation, encapsulation, information hiding, message passing and polymorphism. Students need to have a sound grasp of these basic concepts before they can apply object-oriented technique to solve programming problems. A brief description of these important concepts is presented below based on [6; 22; 39].

An object is a concept, abstraction or thing with known boundaries and meaning for the problem at hand. A class is a concept that describes a set of objects that are specified in the same way. All objects of a given class share a common specification for their features, their semantics and the constraints upon them. In Java programming a class acts as a kind of template from which individual objects are constructed when they are needed. A single object is known as an instance and every object is an instance of some class.

Encapsulation is the placing of data within an object together with the operations that apply to that data, which is the main principle of object-oriented programming approach. The data is stored within the object's attributes and the processes are the object's operations and each has a specific signature. An operation's signature defines the structure and content that a message must have in order to act as a valid call. A signature consists of the name of the operation together with any parameters that the operation needs in order to run it. Encapsulation makes sure that data should only be accessed and altered by operations of the same object. Information hiding states that no object or subsystem

should expose the details of its implementation to other objects or subsystems. Information hiding makes the internal details of an object inaccessible to other objects. Both encapsulation and information hiding compliments to each other and in order to work, objects must exchange messages.

Some operations are specifically written to give access to the data encapsulated within an object. For one object to use the operations of another, it must send a message. Essentially, an object only needs to know its own data and its own operations. However, many processes are complex and require collaboration between objects. The knowledge of some objects must therefore include knowing how to request services from other objects. For another object to access an object's data, it must send a message. When an object receives a message it can tell whether the message is relevant to it. If the message includes a valid signature to one of its operations, the object can respond.

The main use of generalisation in an object-oriented approach is to describe similarity relationships among objects. Inheritance is the mechanism for implementing generalisation and specialisation in an object-oriented programming language. When two classes are related by the mechanism of inheritance, the more general class is called a superclass in relation to the other and the more specialised is called its subclass. A subclass inherits all the characteristics of its superclass and a subclass definition always includes at least one detail not derived from its superclass. In most object-oriented approaches, all classes have a single parent. In some programming languages such as C++, multiple inheritance is also allowed, which means a subclass is a member of more than one hierarchy and inherits from its superclasses in each hierarchy. Overriding means an inherited feature is redefined specific to a subclass.

An advanced object-oriented concept is polymorphism, which refers to the possibility of identical messages being sent to objects of different classes, each of which responds to the message in a different way. It encourages decoupling of subsystems.

4.2 Features to Represent Object-oriented Programming Exercises

In the case of multiple choice questions based assessment tasks, first a topic is selected and then a set of multiple choice questions are designed to assess students ability to understand concepts of the selected topic. A programming exercise is set up to assess student's demonstration of programming concepts and programming abilities. Programming abilities include requirements gathering, designing, coding, testing and deployment. As mentioned earlier, the purpose of modelling student responses to object-oriented programming exercise is to develop basic object-oriented programming abilities for novice programmers. That is why design and coding programming abilities are only considered to be demonstrated in programming exercises.

According to the recommended teaching approach for object-oriented programming as described in the previous section, students need to write a class that represents or models a real world object, concept or idea using a specific object-oriented programming language either Java or C++ after they initially interact with objects and understand and manipulate existing class implementations. One of the object-oriented programming abilities is to write a class that

represents or models a real world object, concept or idea.

4.2.1 Learning Outcomes

To set up a programming exercise that assess object-oriented programming abilities of writing a class, we need to define first appropriate learning outcomes. Defining learning outcomes helps in deciding the scope of the programming exercise, to measure student's achievement level, and generate appropriate feedback. The expected solution to an object-oriented programming exercise is a set of designed and implemented Java or C++ classes. However, for a simple object-oriented programming exercise, which is suitable for beginners, only a design and implementation of one Java or C++ class is enough to solve the problem. The detail of the learning outcomes or objectives to assess student's programming abilities to write a Java class is described below. It includes three components:

1. Object-oriented programming concepts: Students should understand concepts of objects and classes. They should also understand the following concepts: fields, constructors, parameters, data types, access modifiers, return type, variable assignment, accessor and mutator methods and behaviour methods. Finally, students should understand encapsulation and information hiding principles and how to apply them.

2. Object-oriented design abilities: Students should be able to design a simple Java classes based on information given in a problem description, which includes the following design abilities:

   (a) Identifying required fields and specifying their data types based on information given in a problem description.

   (b) Identifying fields initialised to parameter values and fields initialised to default values and specifying data type of each parameter.

   (c) Identifying which fields need to be accessed (get methods) and/or changed (set methods).

   (d) Understanding the purpose of each behaviour method and specifying their requirement based on information given in a problem description and designing an algorithm for each behaviour method. This include identifying the data type of the

required parameters and return type, which fields are updated and/or accessed, specifying local variables and writing instruction steps using logic structures such as conditional statements, case statements and loop statements.

3. Object-oriented implementation or coding abilities: Students should be able to implement simple class definitions properly using Java programming language, which includes the following implementation or coding abilities:

   (a) Declaring fields properly which requires specifying proper field names, access modifiers, and data types.

   (b) Implementing constructors which involve writing constructor headers properly, declaring appropriate parameters and writing proper variable assignment statements to assign fields to parameter or default values.

   (c) Implementing get and set methods properly. Writing get and set methods include two activities. The first step is to specify proper method names, access modifiers, return type and parameter list. Once this step is done, students need to write 'get' and 'set' method headers, return statements and variable assignment statements.

   (d) Implementing the algorithms of each behaviour method specified in the design step. Implementing simple behaviour methods involve different activities. These are declaring local variables, writing variable assignment statements and return statements and using logic structures (if/else, for/while, switch).

4.2.2 Representative Programming Exercise

Once the learning outcomes or objectives are defined, we can set up a programming exercise by first choosing the objects we want to represent and then describe their structure and behaviours. An example of an object-oriented programming exercise is presented in table 4.1. The objective of the exercise is to write a Java class to represent bank account objects. A model solution for the representative object-oriented programming exercise is also shown in appendix C.

A typical student response to the above programming exercise and other related programming exercises is comprised of five main components: declaration of fields, a constructor, get methods, set methods and methods that implement

Table 4.1: Representative programming exercise

Write a class to represent a property for rent from a local letting agent. The class will store the address of the property, the monthly rent, whether the property is occupied or not and the tenants name. The class should have a constructor that will enable all the attributes to be initially set when an instance of the class is created. The constructor requires you to supply, as parameters, the address and the initial monthly rent. For any new property, occupied is set to false and the tenants name is set to an empty string. There should be methods for returning the values of the address, the monthly rent and the tenants name. You are required to write a method that sets the monthly rent to a new value. The method accepts the new monthly rent as a parameter. There should be a method for adding a tenant to a property. The method accepts a new tenants name as a parameter. If the property is not occupied, the tenants name is updated with the parameter input to the method and the occupied status of the property is changed to true. The final method should display all the attributes of a particular property, suitably annotated.

behaviours of objects. In the following paragraphs, features are defined to model student responses to any object-oriented programming exercise whose objective is to write a Java class to represent related objects. The features are categorised into five groups based on the structure of a typical Java class.

The features are defined in such a way that they meet the following criteria optimally. The first criterion is related to training performance of the snap-drift modal learning neural network whose input are student responses represented as a set of defined features. To avoid curse of dimension, the number of features and the possible values for each feature should be as small as possible. The second criterion is the effectiveness of the defined features in supporting the generation of diagnostic feedback that improves student's basic object-oriented programming abilities. The more the number of features and the more specific are the pos-

sible values of each feature, the better support for the generation of diagnostic feedback. The third criterion is the ease of configuring a parsing method that captures the defined features from a student response and convert them into a training pattern, which is represented as a binary vector. Generic features are easier to parse with minimum changes than specific features.

Writing a class properly requires knowledge, understanding and problem solving skills. First, students should understand the concepts of objects and classes. They need to understand that objects represent or model real life things or concepts and are comprised of fields and methods. In contrast, classes describe what related objects have in common in terms of fields and methods. Once a class is defined, it can be used to create objects. Secondly, students should have the ability to understand problem description and map the description into structure of a class. Thirdly, they should understand concepts of fields, methods and constructors. Finally, they should have object-oriented programming abilities of writing classes, declaring fields and implementing methods and constructors.

4.2.3 Correctness of Fields

To understand the concept of fields or attributes of a class, students should know that a class defines related objects in terms of its fields and methods, the different data types, what each data type can represent, the role of fields in a class, and the relationship of fields with methods and constructors. In addition to understanding the concept of field, identifying fields from problem description and assigning appropriate data type for each identified field is very important. Programming abilities concerning fields include declaring fields properly and specifying the cor-

rect access modifier.

Based on the above description of related concepts of field, we can extract the following features: presence of fields, data types of fields, and access modifiers of fields. Presence of fields captures whether the required number of fields are declared or not based on the description of the programming exercise. There are three possible values for this feature. The first one is when all fields are present. The second possible value is when all fields are missing. The third possible value is when one or more fields are missing. It does not specify which fields are missing and the exact number of missed fields in order to simplify the parsing method and reduce the number of possible values.

Data types of fields capture the correctness of data types of declared fields based on the description of the programming exercise. The first possible value is when data types of all declared fields are correct. The second possible value is when the data type of one or more fields are declared properly. These two possible values are applicable when one or more fields are present. Therefore, to address a situation when all fields are missing, a third possible value, which is all fields are missing, is also included.

Access modifiers of fields captures whether fields are declared with correct access modifiers. Based on Java syntax rules, there are three possible access modifiers: public, protected and private. The concepts of encapsulation and information hiding are applied by specifying private access modifiers for fields of a class so that it can not be accessed directly by objects of other classes. Declaring fields as public or protected is not appropriate. Based on these facts, the following possible values are specified: all public or protected, one or more fields declared as public or protected, and all private.

The defined features and their corresponding possible values related to the correctness of fields are presented in table 4.2.

Table 4.2: Features related to correctness of fields

Feature Name	**Possible Values**	**Representation**
Presence of fields	All fields are missing	001
	One or more fields are missing	010
	All fields are present	100
Data types of fields	All fields are missing	001
	Data type of one or more fields are not declared properly	010
	Data type of all fields are declared properly	100
Access modifiers of fields	All fields are missing	0001
	All fields are declared as public or protected	0010
	One or more fields are declared as public or protected	0100
	All fields are declared as private	1000

4.2.4 Correctness of a Constructor

To write a constructor properly, first students need to understand the concept of constructors. Understanding the concept of constructors involves understanding the difference between classes and objects, understanding the concept of fields, knowing the purpose of constructors within a class, and knowing the difference between constructors and instance methods.

In addition to understanding the concept of a constructor, students need to define the requirements of constructors from problem descriptions that include identifying fields which need to be initialised when an object is created. Programming abilities regarding constructors involve writing the header of a constructor properly, specifying the correct data types of all parameters and assigning pa-

rameters to their corresponding fields.

Based on the concept of constructors and its relevant programming abilities, three features are defined related to the correctness of a constructor. They are access modifier of a constructor, return type of a constructor and initialisation of fields. The defined features and their corresponding possible values are presented in table 4.3.

Table 4.3: Features related to correctness of a constructor

Feature Name	**Possible Values**	**Representation**
Access modifier of a constructor	A constructor is missing	001
	Private	010
	Public	100
Return type of a constructor	A constructor is missing	001
	Void or any data type return type	010
	No return type	100
Initialisation of fields	A constructor is missing	0001
	All fields are not initialized at all or initialisation either from parameters or to default values are not done properly	0010
	One or more fields are not initialized at all or initialisation either from parameters or to default values are not done properly	0100
	All fields are properly initialised either from parameters or to default values	1000

4.2.5 Correctness of Methods

Writing methods of a class properly requires understanding of different related concepts. These are concepts of classes, fields, and parameters. Once students understand these concepts, they need to know what a method is and its purpose within a class. They also need to have the ability to specify the requirements of methods of a particular class based on a given problem description. These include

two activities. Firstly, you need to choose appropriate method name, return type, access modifier and parameters for each method. The second activity is to specify the purpose or task of each method using a set of instructions.

Programming abilities regarding methods are writing properly a method header and implementing a specified set of instructions using programming constructs such as sequence, selection, iteration, variable declaration, and variable assignments.

A class defines similar objects in terms of fields and methods. Methods implement behaviours of an object. A class also includes other methods for accessing and changing its fields. These methods are very important as fields can not be accessed and updated directly because of recommended principles of object-oriented programming approach particularly encapsulation and information hide. Methods which return information about the state of an object are called accessor methods or get methods whereas methods that change the state of an object are called mutator methods or set methods [6]. The features defined to capture correctness of get and set methods are presented in table 4.4 and 4.5 respectively.

Table 4.4: Features related to correctness of get methods

Feature Name	**Possible Values**	**Representation**
Presence of get methods	All get methods are missing or incomplete	001
	One or more get methods are missing or incomplete	010
	All get methods are present	100
Access modifier of get methods	All get methods are missing or incomplete	0001
	All private	0010
	One or more get methods are private	0100
	All public	1000
Parameters of get methods	All get methods are missing or incomplete	0001
	All get methods have one or more parameters	0010
	One or more get methods have one or more parameters	0100
	All get methods do not have parameters	1000
Return type of get methods	All get methods are missing or incomplete	0001
	All get methods have no return types or void	0010
	One or more get methods have appropriate return types	0100
	All get methods have appropriate return types	1000
Return statement of get methods	All get methods are missing or incomplete	0001
	All get methods have no return statement or in appropriate returned field name	0010
	One or more get methods have appropriate return statements	0100
	All get methods have appropriate return statements	1000

Table 4.5: Features related to correctness of set methods

Feature Name	**Possible Values**	**Representation**
Presence of set methods	All set methods are missing or incomplete	001
	One or more set methods are missing or incomplete	010
	All set methods are present	100
Access modifier of set methods	All set methods are missing or incomplete	0001
	All private	0010
	One or more set methods are private	0100
	All public	1000
Parameters of set methods	All set methods are missing or incomplete	0001
	All set methods have zero or more than one parameter	0010
	One or more set methods have one parameter	0100
	All set methods have one parameter	1000
Return type of set methods	All set methods are missing or incomplete	0001
	All set methods have return types other than void	0010
	One or more set methods have appropriate return void return types	0100
	All set methods have appropriate void return types	1000
Variable assignment of set methods	All set methods are missing or incomplete	0001
	In all set methods, parameter values are not properly assigned to their corresponding fields	0010
	In one or more set methods, parameter values are properly assigned to their corresponding fields	0100
	In all set methods, parameter values are properly assigned to their corresponding fields	1000

Methods implement behaviour of objects which could be simple or complex depending on the nature of the behaviour. Any object can have one or more methods. Methods that implement behaviours have to be declared properly and the body of methods implement the logic of the behaviours. Most methods change the state of objects which means they update certain fields and they also might

access certain fields. Methods also communicate with other methods to get service. Based on the concept of behaviour methods, we defined generic features that are applicable for any method that implements a behaviour of objects. The defined features are shown in table 4.6.

Table 4.6: Features related to correctness of behaviour methods

Feature Name	**Possible Values**	**Representation**
Access modifier of behaviour methods	All behaviour methods are missing or incomplete	0001
	All private	0010
	One or more behavior methods are public	0100
	All public	1000
Parameters of behaviour method	The behaviour methods is missing or incomplete	0001
	One or more parameters are missing or unnecessary parameters are defined	0010
	Data type of one or more parameters are not as required	0100
	Appropriate parameters are defined	1000
Return type of behaviour method	The behaviour method is missing or incomplete	001
	Return type is not as required	010
	Appropriate return type	100
Status of updated fields	The behaviour method is missing or incomplete	001
	One or more required fields are updated properly	010
	All required fields are updated properly	100
Presence of if statements	The behaviour method is missing or incomplete	001
	One or more required if statements are missing or not defined properly	010
	All required if statements are defined properly	0100

4.3 Parsing Method

Once the features to represent student responses to an object-oriented programming exercise are defined, the challenge is to parse any student response text into a binary vector based on the defined features. The parsing method has to capture each feature correctly and assign an appropriate feature value for all possible student responses. As explained in the previous section, the features are defined in such away that the parsing method can be applied to different programming exercises with minimum changes. The proposed parsing method is described in the following paragraphs.

The parsing method has three input types. The first input is a student response text. The second input is a set of defined features with their possible values and their corresponding representations. The third input is a specification of an object-oriented programming exercise. The specification is done manually by a tutor who designs the object-oriented programming exercise. Since the defined features are not generic, the parsing method needs information about a particular programming exercise in order to capture and score correctly the possible value of each defined feature.

Even though the purpose of the programming exercises is similar, which is to write an object-oriented class that represents similar objects based on the description of the programming exercise, they differ from each other in terms of the number of fields, the data type of each field, whether each field has a default value or being initialised from a constructor, whether each field requires an accessor method or not, and whether each field requires a mutator method or not. In addition to this, programming exercises differ from each other, in terms

of the number of operation methods required to implement the behaviour of the represented objects and the algorithms of each operation method. The details of the algorithm of each operation method differ on how they update fields, which fields they access, and if they use if statements. The use of while or for loops and local variables are not considered as the focus is on simple classes with simple methods for beginners.

4.3.1 Extracting Main Components

As mentioned in the above, the first input of the parsing method is a student response text to an object-oriented programming exercise. The first step of the parsing method is pre-processing the response text, which is a Java source code, to remove comments, trailing spaces, and unnecessary texts. Once this pre-processing step is done, the next step is to extract the main components or break the source code into five main components: fields, constructors, get methods, set methods, and behaviour/operation methods.

These five components are extracted from the source code using a regular expression technique. A regular expression technique enables to represent similar sequence of characters using special symbols. Many programming languages including Java, which is the chosen implementing language, have added regular expression capabilities. The challenge, for example in the case of extracting the field declarations, is to write a search pattern that describes all possible field declarations and should not also include constructor and method declarations as they are similar to a field declaration. The algorithm for extracting the main components using a pseudo-code is described as follows:

1. Set the file path of a student response source code.

2. Transform source code into character sequences.

3. Define a search pattern to represent all possible Java comments:
   `"\\s*/\\*.*|\\s*\\*.*|\\s*//.*|\\s*\\*/"`

4. Remove characters that match the search pattern.

5. Retrieve the class name using a search pattern:
   `"\\b[Pp]ublic\\b\\s+\\b[Cc]lass\\b\\s+\\w+"`

6. Store the class name to a variable.

7. Define a search pattern to represent all possible Java field declarations:
   `"\\b([Pp]ublic|[Pp]rivate|[Pp]rotected)\\b\\s+\\w+\\s+\\w+\\s*;"`

8. Search the character sequences using the defined pattern to retrieve all field declarations.

9. Define a search pattern to represent all possible Java constructor declaration and body of a constructor:
   ```
   "\\b([Pp]ublic|[Pp]rivate|[Pp]rotected)\\b\\s+\\b" + class name
    + "\\b\\s*\\(.*\\)\\s*\\{(\\s*.+=.+|;)*\\s*\\}"
   ```

10. Search the character sequences using the defined pattern to retrieve all constructors.

11. Define a search pattern to represent all possible method declarations and method body of get methods:
    ```
    "\\b([Pp]ublic|[Pp]rivate|[Pp]rotected)\\b.+\\(.*\\)\\s+\\
     {(\\s*return\\s*.*|;){1}\\s*\\}";"
    ```

12. Search the character sequences using the defined pattern and retrieve all get methods.

13. Define a search pattern to represent all possible method declarations and method body of set methods:
    ```
    "\\b([Pp]ublic|[Pp]rivate|[Pp]rotected)\\b\\s*void.+\\
    (.*\\)\\s+\\{(\\s*.+=.+|;){1}\\s*\\}"
    ```

14. Search the character sequences using the defined pattern and retrieve all set methods.

15. Define a search pattern to represent all possible behaviour/operation method declarations and method body :
    `"\\b([Pp]ublic|[Pp]rivate|[Pp]rotected)\\b.+\\(.*\\)"`

16. Search the character sequences using the defined pattern and retrieve all behaviour methods.

17. Output a collection of field declarations, a collection of constructors, a collection of get methods, a collection of set methods, and a collection of behaviour methods.

4.3.2 Capture and Score Feature Values

Once the five main components are extracted, the next steps are to capture and score the correct values of all defined features based on the outputs of the above algorithm. In the following sections, how each defined feature is captured and scored will be described. The defined features are organised based on the five main components of a student response text, that is why they are described in the following five sections.

4.3.2.1 Correctness of Fields

Three features are defined to check the correctness of fields: presence of fields, data types of fields and access modifiers of fields. To capture and score the appropriate values of these features for a given student response text, the parsing method first scans each extracted field declaration in order to determine the number of field declarations with Char, String, Numeric and Boolean data types. It also determines the number of field declarations with Private and Public/Protected access modifiers. Secondly, the parsing method compares the information about the field declarations with the programming specification, in order to determine the correct feature values of the three features related to the correctness of fields.

4.3.2.2 Correctness of a Constructor

The three features which are defined to check the correctness of constructors are access modifier of a constructor, return type of a constructor and initialisation of fields. Student responses are not expected to have more than one constructor as they are required to write a simple Java class. The algorithm to capture and score the correct feature values of these three features related to the correctness of a constructor is described using pseudo-code as follows:

1. Get extracted constructor.
2. Define a search pattern to find the access modifier:
   `"\\b([Pp]ublic|[Pp]rotected)\\b"`
3. Record the matched access modifier.
4. Define a search pattern to find the return type:
   `"\\b(void|String|string|char|int|double|float|boolean)\\b.+\\("`
5. Record the matched return type.
6. Extract parameter list and record it.
7. Extract and record field names from the field declarations.
8. Extract assignment statements defined within the constructor:
   `"\\s*" + field name + "\\s*=.*"`
9. Record the extracted assignment statements.
10. Check each extracted assignment statement whether each field name is initialised properly to one of the parameter or to a default value.
11. Score the appropriate feature values using the recorded information and syntax rules of Java programming to declare and write a constructor.

4.3.2.3 Correctness of Get Methods

Five features are defined to check the correctness of get methods. These features are presence of get methods, access modifiers of get methods, parameters of get methods, return type of get methods, and return statements of get methods. The algorithm to capture and score the correct values of these features is described using pseudo-code as follows:

1. Get extracted get methods.
2. Define a search pattern to find the access modifier of each get method:
   `"\\b([Pp]ublic|[Pp]rotected)\\b" or`
   `"\\b[Pp]rivate\\b"`
3. Record the number of private and public access modifiers.
4. Define a search pattern to find the return type of each get method:
   `"\\b(String|string|char|int|double|float|boolean)\\b.+\\(" or`
   `"\\b[Vv]oid\\b.+\\"`
5. Record the number of get methods with a numeric,char, string or char return types.
6. Record the number of get methods with void return types.
7. Record the number of missed or incomplete get methods based on the specification of the programming exercise.
8. Extract parameter list of each get method and record them.
9. Record the number of get methods with no parameters and with one or more parameters.
10. Extract and record field names from the field declarations.
11. Extract return statements defined within each get method:
    `"(return|Return)(\\s*)" + field name`
12. Record the extracted return statements.
13. Check each extracted return statement whether one of the declared field names is returned properly or not.
14. Record the number of get methods with no return statements.

15. Record the number of get methods with appropriate return statements.

16. Score the appropriate feature values using the recorded information and syntax rules of Java programming to declare and write a get method.

4.3.2.4 Correctness of Set Methods

Five features are defined to check the correctness of set methods. These features are presence of set methods, access modifiers of set methods, parameters of set methods, return type of set methods, and variable assignments of set methods. The algorithm to capture and score the correct values of these features is described using pseudo-code as follows:

1. Get extracted set methods.

2. Define a search pattern to find the access modifier of each set method:
   `"\\b([Pp]ublic|[Pp]rotected)\\b" or`
   `"\\b[Pp]rivate\\b"`

3. Record the number of private and public access modifiers.

4. Define a search pattern to find the return type of each set method:
   `"\\b(String|string|char|int|double|float|boolean)\\b.+\\(" or`
   `"\\b[Vv]oid\\b.+\\"`

5. Record the number of set methods with a numeric,char, string or char return types.

6. Record the number of set methods with void return types.

7. Record the number of missed or incomplete set methods based on the specification of the programming exercise.

8. Extract parameter list of each set method and record them.

9. Record the number of set methods with one parameter.

10. Record the number of set methods with no parameter or more than one parameter.

11. Extract and record field names from the field declarations.

12. Extract variable statements defined within each set method:
    `"\\s*" + field name + "\\s*=.+"`
13. Record the extracted variable statements.
14. Check each extracted variable statement whether parameter values are properly assigned to their corresponding fields.
15. Record the number of set methods with appropriate variable statements.
16. Score the appropriate feature values using the recorded information and syntax rules of Java programming to declare and write a set method.

4.3.2.5 Correctness of a Behaviour Method

The correctness of a behaviour method is checked using six features, which are access modifier of a behaviour method, parameters of a behaviour method, return type of a behaviour method, status of updated fields of a behaviour method, status of accessed fields of a behaviour method, and presence of IF statements. The last three features are optional since they are not applicable to any behaviour method. As mentioned earlier, the specification of the programming exercise under consideration holds the information whether these features are applicable and other details such as which fields or updated or accessed. The algorithm to capture and score the correct values of these features is described using pseudo-code as follows:

1. Get extracted behaviour methods.
2. For each behaviour method do:
    (a) Define a search pattern to find the access modifier of the current behaviour method:
    `"\\b([Pp]ublic|[Pp]rotected)\\b" or`
    `"\\b[Pp]rivate\\b"`
    (b) Record the matched access modifier.

(c) Define a search pattern to find the return type of the current behaviour method: `"\\b(String|string|char|int|double|float|boolean)\\b.+\\(" or "\\b[Vv]oid\\b.+\\"`

(d) Record the matched return type.

(e) Extract parameter list of the current behaviour method.

(f) Compute the number of Char parameter types.

(g) Compute the number of String parameter types.

(h) Compute the number of Numeric parameter types.

(i) Compute the number of Boolean parameter types.

(j) If the current behaviour method has updated fields THEN

i. Check if each required field is updated properly.

ii. Record the number of fields updated properly.

(k) If the current behaviour method has accessed fields THEN

i. Check if each required field is accessed properly.

ii. Record the number of fields accessed properly.

(l) If the current behaviour method has if statements THEN

i. Check if each required IF statement is defined properly.

ii. Record the number of IF statements defined properly.

3. END FOR EACH

4. Score the appropriate feature values using the recorded information, specification of the programming exercise, and syntax rules of Java programming to declare and write a behaviour/operation method.

4.4 Develop Student Group Profiles

In the previous sections, features have been defined to represent student responses to an object-oriented programming exercise. Furthermore, a parsing method is proposed to extract, code and transform the defined features into binary vector from a student response text. The focus of this section is to explain how a snap-drift modal learning neural network can be applied to identify student group

profiles that represent different basic object-oriented programming abilities based on the defined features and on the proposed parsing method. The purpose of identifying student group profiles is to construct diagnostic feedback appropriate for each student group that improves basic object-oriented programming abilities.

4.4.1 Training Data Set

The representative programming exercise described in section 4.2.2 is used to gather student responses in order to prepare a training data set. All students registered for Introduction to Java programming module were invited to undertake a formative assessment. A web-based formative assessment tool was designed and developed to conduct an assessment session. The software tool enables to set up a programming exercise, manage, and store them. In addition to this, the tool includes a front end user interface to display the programming exercise description and provide an empty text area for students to write their programing solution.

Students weren't allowed to copy from each other during a trial. Once a student submits the attempted solution, the student response is stored in a single file named using student's user name. Seventy two students participated in the trial, however only 59 student responses were considered as training patterns due to quality issues. The screen shot of the front end of the web-based formative assessment tool is attached in appendix D.

4.4.2 Snap-drift Modal Learning Neural Network

In chapter three, a snap-drift modal learning neural network was designed and implemented using Java programming language. The developed tool includes also an interactive user interface to choose learning parameters and display the identified student group profiles. As mentioned in chapter three, the software tool supports the application of snap-drift modal learning neural networks to identify student group profiles based on student responses to multiple choice questions.

To be able to identify student group profiles based on student responses to an object-oriented programming exercise, the software tool needs to be extended by designing and developing a data structure to define and manage the proposed features and programming specification described in the previous sections. In addition to this, the proposed parsing method was also implemented and integrated with the software tool. A screen shot of the extended software tool is depicted in figure 4.1.

As shown in the training data panel of the user interface depicted in figure 4.1, the training patterns are binary vectors, which are the output of the parsing method that captures and transforms student response texts. In the output panel, identified student group profiles are displayed. The rows of the table represent the defined features, whereas each column specifies the most likely value of for each student group. The profiling method proposed and applied in chapter three is extended to profile the identified student groups.

To identify useful student group profiles that facilitate effective diagnostic feedback, different combinations of the learning parameters were searched using the interactive user interface that allows to change each learning parameter easily.

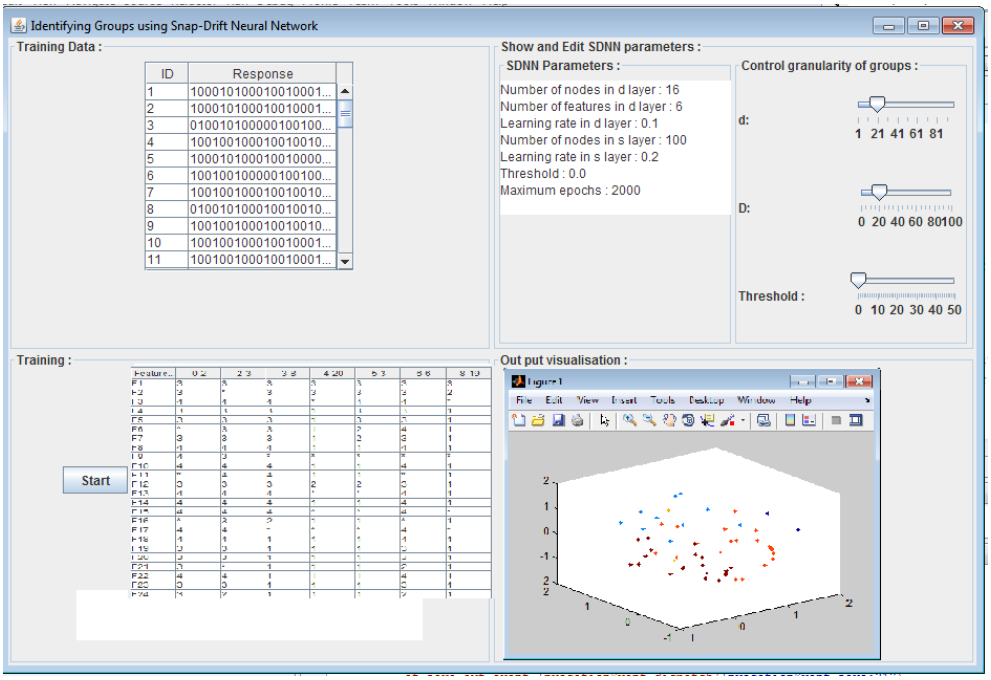


Figure 4.1: Screen shot of Graphical user interface of SDNN tool

The criteria for assessing usefulness of a particular set of student group profiles was adopted from chapter three. The two defined criteria are the characteristic of each student group and the number of student groups. In addition to these two criteria, a third criterion was defined, which is the number of asterisk (*) in each row of the table that displays the identified student group profiles. The usefulness of a set of student group profiles increases as the number of rows with all column values of asterisk (*) decreases.

The student group profiles depicted in figure 4.2 was identified using the learning parameters shown in table 4.7.

Feature-Name	0-2	2-3	3-8	4-20	5-3	6-6	8-19
F1	3	3	3	3	3	3	3
F2	3	*	3	3	3	3	2
F3	4	4	4	*	4	4	*
F4	3	3	3	1	3	3	1
F5	3	3	3	1	3	3	1
F6	*	3	3	1	2	4	1
F7	3	3	3	1	2	3	1
F8	4	4	4	1	1	4	1
F9	4	3	*	*	*	*	*
F10	4	4	4	1	1	4	1
F11	*	4	4	1	1	*	1
F12	3	3	3	2	2	3	1
F13	4	4	4	*	*	4	1
F14	4	4	4	1	1	4	1
F15	4	4	4	*	*	4	*
F16	*	3	2	1	1	*	1
F17	4	4	*	*	*	4	*
F18	4	4	1	1	1	4	1
F19	3	3	1	1	1	3	1
F20	3	3	1	1	1	3	1
F21	3	*	1	1	1	2	1
F22	4	4	1	1	1	4	1
F23	3	3	1	1	1	3	1
F24	3	2	1	1	1	2	1

Figure 4.2: Student group profiles. The column header (n-m) specifies winning node (n) and number of student responses (m)

Table 4.7: Snap-drift modal learning neural network learning parameters

Learning Parameter	**Value**	**Learning Parameter**	**Value**
Maximum epoch	5000	Minimum epoch	1000
Convergence percentage	95	Group profile threshold	75
d	25	D	10
s	20	Threshold	2.5

4.5 Summary

The main objective of this chapter is investigating the application of a snap-drift modal learning neural network to model student responses gathered from

programming exercise. The purpose of modelling student responses to an object-oriented programming exercise is identifying student group profiles that represent different programming abilities of writing an object-oriented class.

Firstly, a brief literature review on object-oriented programming approach was conducted. Based on the review, most Universities are adopting object-oriented programming approach to teach first year studying Computer Science. The main reason for adopting object-oriented programming approach is the fact that it supports top-down teaching approach, which is the recommended teaching approach. The other reason is the popularity and application of object-oriented programming languages in several industries.

The main difference between procedural languages such as Pascal, Basic, and C and object-oriented programming approach is the relationship between an algorithm and the data it operates on. In the case of procedural languages there is a clear distinction between an algorithm and the data it operates on. However, in the case of object-oriented programming approach the data is encapsulated with the functions that act upon it.

They also differ in their teaching approaches. Procedural languages apply a bottom-up approach, which starts with the basics or fundamental of programming and then slowly builds knowledge and programming skill over time. However, object-oriented programming supports top-down teaching approach, which is also known as object first. Students initially interact with sample programs to familiarise with specific object-oriented programming features, then they can start learning how an object is implemented, modified or extended using a specific programming language such as Java or C++.

Secondly, features were defined to represent object-oriented programming ex-

ercises. To define features, first what object-oriented concepts and what programming abilities to assess were specified. The next step was to set up a representative programming exercise and define criteria.

The first criterion is related to training performance of the snap-drift modal learning neural network whose input are student responses represented as a set of defined features. The second criterion is the effectiveness of the defined features in supporting the generation of diagnostic feedback that improves student's basic object-oriented programming abilities. The third criterion is the ease of configuring a parsing method that captures the defined features from a student response and convert them into a training pattern, which is represented as a binary vector. The features were defined in such a way that they meet the three defined criteria.

Once the features to represent student responses to an object-oriented programming exercise are defined, a parsing method was proposed to parse any student response text into a binary vector. The parsing method has to capture each feature correctly and assign an appropriate feature value for all possible student responses. The parsing method has three input types. The first input is a student response text. The second input is a set of defined features with their possible values and their corresponding representations. The third input is a specification of an object-oriented programming exercise.

Thirdly, snap-drift modal learning neural network was applied to identify student group profiles that represent different basic object-oriented programming abilities based on the defined features and on the proposed parsing method. The purpose of identifying student group profiles is to construct diagnostic feedback appropriate for each student group that improves basic object-oriented programming abilities. First training data were gathered based on the representative

programing exercise by conducting assessment sessions. Seventy two students participated in the trial, however only 59 student responses were considered as training patterns due to quality issues.

Once a training data set was prepared, the snap-drift modal learning neural network tool that was developed in the previous chapter, was extended in order to be able to identify student group profiles based on student responses to an object-oriented programming exercise. It was extended by designing and developing a data structure to define and manage the proposed features and programming specification described in the previous sections. In addition to this, the proposed parsing method was also implemented and integrated with the software tool.

Finally, useful student group profiles were identified that facilitate effective diagnostic feedback. Different combinations of the learning parameters were searched using an interactive graphical user interface in order to find useful student group profiles. The criteria for assessing usefulness of a particular set of student group profiles was adopted from chapter three. The two defined criteria are the characteristic of each student group and the number of student groups. In addition to these two criteria, a third criterion was defined, which is the number of asterisk (*) in each row of the table that displays the identified student group profiles. Based on the defined criteria, seven student group profiles were identified that clearly represent different object-oriented programming abilities.

Chapter 5

Evaluation of Student Group Profiles

In the previous chapter, the application of a snap-drift modal learning neural network was extended to model student responses gathered from programming exercise. The purpose of modelling student responses to an object-oriented programming exercise is to identify student group profiles that represent different programming abilities of writing an object-oriented class.

The main purpose of this chapter is to evaluate the proposed method for identifying student group profiles that facilitates generation of diagnostic feedback to improve development of basic object-oriented programming abilities. First, a diagnostic feedback was constructed for each identified student group. Secondly, trials were conducted to gather student responses. Thirdly, data was prepared by extracting raw data captured by a web-based formative assessment tool. Fourthly, a qualitative analysis of individual cases was performed. Fifthly, the assessment of whether the constructed diagnostic feedback improves the overall learning per-

formance was undertaken. Finally, further improvements on the identified weaknesses of the method are proposed.

5.1 Construct Diagnostic Feedback

A procedure was proposed in chapter three to construct diagnostic feedback based on identified student group profiles. To construct diagnostic feedback for the identified student group profiles, the learning outcomes described in section 4.2.1 and the patterns in each student group profile are analysed.

Table 5.1: Diagnostic feedback for student group one

Your solution is not complete. A complete Java class should contain three main components: fields,a constructor and methods. Your solution includes only fields. First of all, make sure all required fields are declared and choose the right data type from the possible types (String, char, int, boolean). Once all required fields are defined, they need to be initialised properly using a constructor. A constructor is a special method whose name is the same as its class name and doesn't have a return type. In addition to fields and a constructor, you need to write all required methods: get methods, set methods and methods that implement behaviours of property objects. Read the problem description carefully to identify which fields need get and set methods and what operations can be done to property objects.

The patterns in each student group profile is used to identify the current levels of object-oriented programming abilities and the learning outcomes is used to define the expected level of object-oriented programming abilities. Once the current levels and gaps of object-oriented programming abilities are identified for each student group profile, a diagnostic feedback is constructed that closes

the gaps and improves the object-oriented programming abilities. The set of constructed diagnostic feedback for each student group profile shown in figure 4.2 are shown in tables 5.1 to 5.7.

Table 5.1 shows the diagnostic feedback for the first student group. Students who belong to this group have very low levels of programming abilities. They seem to understand the concept of fields, however they don't know how to write fields properly.

Table 5.2: Diagnostic feedback for student group two

Your solution is not complete. A complete Java class should contain three main components: fields, a constructor and methods. You have declared the required fields properly. Once all required fields are defined, they need to be initialised properly using a constructor. A constructor is a special method whose name is the same as its class name and doesn't have a return type. In addition to fields and a constructor, you need to write all required methods: get methods, set methods and methods that implement behaviours of property objects. Read the problem description carefully to identify which fields need get and set methods and what operations can be done to property objects.

Table 5.2 shows the diagnostic feedback for the second student group. Students who belong to this group have low levels of programming abilities, which is slightly better than students who belong to the first group. They understand the concepts of fields and know how to write fields properly.

Table 5.3: Diagnostic feedback for student group three

You have done well in defining the fields properly and including a constructor. However, your solution is not complete. First of all, make sure that all fields are initialised properly using a constructor. All required methods for retrieving and changing values of fields and implementing behaviours of property objects are missing or incomplete. A get method returns the value of one of the objects field. It contains a return statement and does not require a parameter. Set methods have void return type and one formal parameter. They have only one variable assignment statement. Read carefully the problem description in order to identify all required behaviours of property objects. Once they are identified, specify clearly how they are implemented using methods. Usually methods update fields to change state of objects. It is important to identify which fields need to be updated. Assignment statements are used to update values of fields. Field name = new value;

Table 5.3 shows the diagnostic feedback for the third student group. Students who belong to this group understand the concepts of fields and know how to write fields properly. They also understand the concept of constructors and know how to write constructors. They don't know how to initialise fields properly using a constructor, and how to design and implement methods.

Table 5.4: Diagnostic feedback for student group four

You have done well in defining the fields properly and initialising them using a constructor. However, your solution is not complete. All required methods that implement behaviours of property objects are missing or incomplete. Read carefully the problem description in order to identify all required behaviours of property objects. Once they are identified, specify clearly how they are implemented using methods. Usually methods access values of fields or update them to change state of objects. It is important to identify which fields need to be accessed or updated. Assignment statements are used to update values of fields. Field name = new value; This an example of an assignment statement, the new value to be assigned is on the right hand side. To access a value of a field use its field name without a quotation. System.out.println(balance); In this Java statement, a field(variable) named balance is accessed to be displayed.

Table 5.4 shows the diagnostic feedback for the fourth student group. Students who belong to this group understand the concepts of fields and know how to write fields properly. They also understand the concept of constructors, know how to write constructors, and initialise fields properly using a constructor. However, they don't know how to design and implement methods.

Table 5.5: Diagnostic feedback for student group five

Overall your solution includes the three main components of a class: fields, a constructor and methods. However, there are minor errors in your solution. First of all, make sure all fields are initialised properly using a constructor. Read carefully the problem description in order to specify clearly what happens when a method is executed. Usually methods access values of fields or update them to change state of objects. It is important to identify which fields need to be accessed or updated. Assignment statements are used to update values of fields. fieldname = newvalue; This an example of an assignment statement, the new value to be assigned is on the right hand side. To access a value of a field use its field name without a quotation. System.out.println(balance); In this Java statement, a field(variable) named balance is accessed to be displayed.

Table 5.5 shows the diagnostic feedback for the fifth student group. Students who belong to this group understand the concepts of fields, constructors and methods. They also know how to write a simple class with minor errors.

Table 5.6: Diagnostic feedback for student group six

Overall your solution includes the three main components of a class: fields, a constructor and methods. However, you need to check carefully the implementation of your methods and check whether all required get and set methods are included. Read carefully the problem description in order to specify clearly what happens when a method is executed. Are the statements executed in sequence or do you need conditional statements. Try to write the algorithm of a method using pseudo-code before you start implementing it using Java.

Table 5.6 shows the diagnostic feedback for the sixth student group. Students

who belong to this group understand the concepts of fields, constructors and methods. They also know how to write a class. They lack the programming ability of designing and implementing correct algorithms of methods based on the problem description.

Table 5.7: Diagnostic feedback for student group seven

Well done. All components of the property class are implemented correctly. All fields are declared properly and initialised correctly using a constructor. All get and set methods are implemented correctly. All specified behaviours of property objects are implemented correctly using add tenant and display methods. You could improve the programming style of your Java class: proper names of fields and methods, adding comment description to every method and having proper spaces and indentations. Unfortunately this web based system doesn't have compiling and testing features. If you want to compile and test your program please use BlueJ.

Table 5.7 shows the diagnostic feedback for the seventh student group. Students who belong to this group understand the concepts of fields, constructors and methods. They also know how to write a class properly.

5.2 Assessment Session

To gather student responses to an object-oriented programming exercise, assessment sessions were conducted. A web-based formative assessment tool was designed and implemented to conduct assessment sessions so that students are able to undertake assessment tasks based on programming exercises, receive a diagnostic feedback based on their responses instantly, attempt multiple times and

record information about the session.

The web-based software tool integrates five components, which are the implemented snap-drift modal learning neural network, assessment tasks based on programming exercises, diagnostic feedback, assessment session manager and a relational database. The tool was implemented using web technologies (XHTML, CSS, Java script and AJAX), Java technologies (JSP and Servlet), MySql relational database server and Hibernate for mapping relational tables to Java objects.

A trial was conducted using the developed web-based formative assessment tool. During the trial, students were not allowed to copy from each other. A different cohort of students, who didn't participate during a training phase, were chosen to take part in the trial. All participated students were registered for the Introduction to Programming module offered to first year students enrolled for Computer Science course at London Metropolitan University. Thirty five students participated during the first trial.

5.3 Data Preparation

During the trial, raw data was captured by the web-based formative software tool for each student that undertakes the assessment session. Data preparation activities were required before conducting analysis and evaluation to make it suitable and to clean the data by removing missing, unreadable or incomplete student responses.

The raw data were stored in text based files and contain the student response source code for each attempt. In addition to this, the raw data contain informa-

tion about all attempts per each student. The information reveals the attempt number, the duration, and the assigned winning node. The assigned winning node for each attempt is used to determine a student group profile. Knowing the student group profile of a student attempt is important as it reveals the diagnostic feedback for the student attempt.

The first step includes downloading generated files and then checking manually the quality for missing, unreadable files or incomplete data. The second step was to link student source codes into their corresponding attempt number and assign a group number for each attempt based on the assigned winning node. Finally, we scored the value of each defined feature for all attempts using automated and manual marking. The automated marking was done based on the proposed and developed parsing method.

Out of the thirty five student participants, only twenty five students were considered because of quality issues. The distribution of the number of attempts for the twenty five students is depicted in figure 5.1. The minimum and maximum number of attempts are 1 and 11 respectively. The average number of attempts is 4.16.

5.4 Qualitative Analysis of Individual Cases

Based on the data prepared in the previous section, a qualitative analysis of individual cases was conducted. The objective of the analysis is evaluation of the performance of the parsing method and assessment of the effectiveness of the constructed diagnostic feedback.

In the following sections, the analysis of fifteen participants out of the twenty

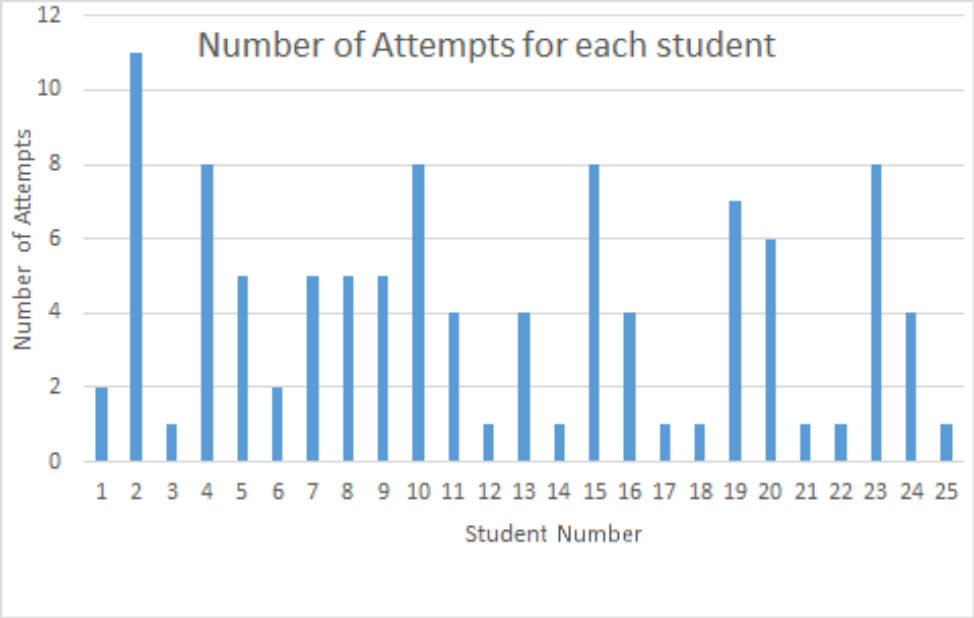


Figure 5.1: The distribution of the number of student attempts

five students is described. Only students who had attempted more than two were considered.

Student number 2 as shown in table 5.8 attempted eleven times. The student remained in the same state or group, which indicates levels of programming abilities, for the first eight attempts and then transferred into another state in the ninth attempt and remained in that state for the following three attempts.

Table 5.8: The sequence of attempts for student number 2

Attempt No.	**Group Number**	**Attempt No.**	**Group Number**
1	1	7	1
2	1	8	1
3	1	9	2
4	1	10	2
5	1	11	2
6	1		

The initial levels of programming abilities for student number 2 were low as

the first attempt showed which includes only a partial declaration of fields and a constructor. During the course of the eleven attempts, the student improved his/her levels of programming abilities as he/she transferred into a different state, which is student group two.

The analysis of the Java source codes for each student attempt showed that student number 2 improved his/her understanding of set and get methods and programming abilities in writing a header of constructor and set methods.

As mentioned earlier, student number 2 remained in the same state for eight attempts. The student received a diagnostic feedback for student group one during these eight attempts. As shown in table 5.1, the diagnostic feedback was appropriate to improve his/her levels of programming abilities. However, it took him/her eight attempts to transfer into another student group. The comparison between the automated and manual marking showed that the parsing method could not capture incomplete constructor and set methods. This could be one possible reason why student number 1 remained in the same state for the first eight attempts.

Table 5.9: The sequence of attempts for student number 4

Attempt No.	**Group Number**	**Attempt No.**	**Group Number**
1	1	5	2
2	2	6	3
3	2	7	3
4	2	8	3

Student number 4 as shown in table 5.9 attempted eight times. After the first attempt the student transferred into another state, which is student group two. Finally, the student transferred into student group three after four attempts and remained in student group three state for the following three attempts.

The initial levels of programming abilities for student number 4 were very low as the first attempt showed which includes only an incomplete declaration of fields and incorrect codes for writing set and get methods. During the course of the eight attempts, the student improved his/her levels of programming abilities as he/she transferred into a different state, which is student group two and then into student group three.

The analysis of the Java source codes for each student attempt showed that student number 4 improved his/her understanding of set and get methods and programming abilities in writing a constructor, get methods and set methods. This indicates that the diagnostic feedback received by the student had a positive impact in improving his/her programming abilities.

Table 5.10: The sequence of attempts for student number 5 and 7

Attempt No.	**Group Number**
1	1
2	1
3	1
4	3
5	3

Student number 5 and 7 as shown in table 5.10 attempted five times. The students remained in the same state or group, which indicates levels of programming abilities, for the first three attempts and then transferred into a higher state in the fourth attempt and remained in that state in the final attempt.

The initial levels of programming abilities for student number 5 and 7 were very low as shown in their first attempts. The first attempt of student number 5 included only an incomplete declaration of fields and a constructor. The first attempt of student number 7 included only a proper declaration of fields but not

all required fields were identified. During the course of the five attempts, both students improved their levels of programming abilities as they transferred into a different state, which is student group three.

The analysis of the Java source codes for each student attempt showed that student number 5 improved his/her understanding of fields and constructors and programming abilities in writing fields and constructors. The same analysis for student number 7 showed that he/she improved understanding of constructors and programming abilities in writing constructors.

As mentioned earlier, student number 5 and 7 remained in the same state for the first three attempts. Both students received a diagnostic feedback for student group one during these three attempts. As shown in table 5.1, the diagnostic feedback was appropriate to improve their levels of programming abilities. As the source codes of the second and third attempts for both students showed, Both students improved slightly their programming abilities during these attempts. The reason for remaining in the same state was due to the fact that the improvements were not adequate enough to transfer the student into a higher state. Both students managed to transfer into a higher state at the fourth attempt, which indicated the importance of the received diagnostic feedback.

Table 5.11: The sequence of attempts for student number 8

Attempt No.	**Group Number**
1	4
2	4
3	4
4	4
5	4

Student number 8 as shown in table 5.11 attempted five times. The stu-

dent remained in the same state or group, which indicates levels of programming abilities, for all five attempts.

The initial levels of programming abilities for student number 8 were very good as shown in the first attempt. It includes a complete declaration of fields, a constructor, get and set methods and behaviour methods. Based on this information, the student should be assigned to student group six instead of student group four.

Comparison of automated and manual marking showed that the parsing method could not capture the get methods and behaviour methods correctly, which explains why the student was not assigned to an appropriate student group and why the student did not receive appropriate or useful diagnostic feedback as shown in table 5.4.

Table 5.12: The sequence of attempts for student number 9

Attempt No.	**Group Number**
1	3
2	3
3	3
4	4
5	6

Student number 9 as shown in table 5.12 attempted five times. The student remained in the same state, which is student group three, during the first three attempts. In the fourth attempt, the student transferred into another state, which is student group four. Finally, the student transferred into student group six in the last attempt.

The initial levels of programming abilities for student number 9 were good as the first attempt showed which includes all required components of a class:

fields, a constructor, get and set methods and behaviour methods. Based on this information, the student should be assigned to student group six instead of student group three. The comparison between the automated and manual marking showed that the parsing method could not capture all implemented methods as the method names were different from what the parsing method expected. That is why the the first three attempts were assigned to student group three instead of student group six.

The analysis of the Java source codes for each student attempt showed that student have tried to modify the source code particularly names of the implemented methods. That is why the student transferred into student group four in his/her fourth attempt and then transferred into student group six in the last attempt.

Table 5.13: The sequence of attempts for student number 10

Attempt No.	**Group Number**	**Attempt No.**	**Group Number**
1	3	5	4
2	3	6	4
3	3	7	6
4	3	8	6

Based on the comparison of the initial levels of programming abilities during the first attempt and the levels at the last attempt, student number 10 did not improve, particularly in identifying required all get methods, initialising properly all fields using a constructor, and specifying appropriate parameters of behaviour methods. The diagnostic feedback given during the first three attempts as shown in table 5.3 and the other feedback given in the fourth attempt as shown in table 5.4 were not useful. The most likely reason was as explained earlier due to the inaccurate scoring of the defined features by the parsing method which results in

the assignment of wrong student group.

Student number 10 attempted eight times. The student remained in the same state, which is student group two, during the first four attempts. The student transferred into student group four in the fifth attempt and then transferred into student group six in the seventh attempt and remained in that state until the last attempt.

The initial levels of programming abilities for student number 10 were fair as shown in the first attempt. It includes a correct declaration of all required fields, a constructor and all required get methods. During the course of the eight attempts, the student improved his/her levels of programming abilities as they transferred into higher states, first into the student group four after fifth attempt and then into the student group six after the seventh attempt.

The analysis of the Java source codes for each student attempt showed that student number 10 improved his/her understanding of set methods and behaviour methods and programming abilities in writing set methods and behaviour methods. The gradual improvements of student number 10 indicated the appropriateness or usefulness of the diagnostic feedback given at each attempt.

Table 5.14: The sequence of attempts for student number 11 and 13

Attempt No.	**Group Number**
1	1
2	1
3	1
4	1

Student number 11 and 13 as shown in table 5.14 attempted four times. The students remained in the same state or group, which indicates levels of programming abilities, during all attempts.

The initial levels of programming abilities for student number 11 and 13 were very low as shown in their first attempts, which include only an incomplete declaration of fields. During the course of the four attempts, both students did not improve their levels of programming abilities even though an appropriate or useful diagnostic feedback was provided as shown in table 5.1.

Table 5.15: The sequence of attempts for student number 15

Attempt No.	**Group Number**	**Attempt No.**	**Group Number**
1	2	5	4
2	2	6	4
3	2	7	4
4	2	8	4

Student number 15 as shown in table 5.15 attempted eight times. The student remained in the same state, which is student group two, during the first four attempts. In the fifth attempt, the student transferred into another state, which is student group four and remained in that state until the last attempt.

The initial levels of programming abilities for student number 15 were fair as the first attempt showed which includes proper declaration of all required fields, a constructor, get methods and set methods with minor syntax errors. Based on this information, the student should be assigned to student group four instead of student group two. The comparison between the automated and manual marking showed that the parsing method could not capture the implemented constructor because of unexpected naming of the constructor that differs from the class name. That is why the the first four attempts were assigned to student group two instead of student group four.

Based on the comparison of the initial levels of programming abilities during the first attempt and the levels at the last attempt, student number 15 showed

small improvement, particularly in initialising fields using a constructor and syntax rules for writing a constructor, writing get methods, and writing set methods. Even though the diagnostic feedback given during the first four attempts as shown in table 5.2 were not useful because of inaccurate scoring of features by the parsing method, an appropriate or useful diagnostic feedback was given during the last four attempts. However, the student could not transfer into a higher state as the improvements were not enough.

Table 5.16: The sequence of attempts for student number 16

Attempt No.	**Group Number**
1	3
2	3
3	4
4	4

Student number 16 as shown in table 5.16 attempted four times. The students remained in the same state or group, which indicates levels of programming abilities, for the first two attempts and then transferred into a higher state in the third attempt and remained in that state in the final attempt.

The initial levels of programming abilities for student number 16 were fair as shown in the first attempt, which includes a complete declaration of fields and constructor, initialisation of fields by a constructor, and set methods. During the course of the four attempts, the student improved his/her levels of programming abilities as they transferred into a higher state, which is student group four.

The analysis of the Java source codes for each student attempt showed that the student improved his/her understanding of get methods and programming abilities in writing get methods. This improvements are not as expected considering the diagnostic feedback as shown in table 5.4 given during the third and

fourth attempts.

Table 5.17: The sequence of attempts for student number 19

Attempt No.	**Group Number**	**Attempt No.**	**Group Number**
1	2	5	2
2	2	6	2
3	2	7	4
4	2		

Student number 19 as shown in table 5.17 attempted seven times. The student remained in the same state, which is student group two, during the first six attempts. In the last attempt, the student transferred into another state, which is student group four.

The initial levels of programming abilities of student number 19 were fair as the first attempt showed which includes proper declaration of all required fields, a constructor, get methods and set methods with minor syntax errors. Based on this information, the student should be assigned to student group four instead of student group two. The comparison between the automated and manual marking showed that the parsing method could not capture the implemented constructor because of unexpected naming of the constructor that differs from the class name. That is why the the first six attempts were assigned to student group two instead of student group four.

Based on the comparison of the initial levels of programming abilities during the first attempt and the levels at the last attempt, student number 19 showed small improvement, particularly in initialising fields using a constructor and syntax rules for writing a constructor, writing get methods, and writing set methods. The most likely reason why student number 19 did not improve as expected during the course of the seven attempts is due to the fact that the diagnostic feedback

given during the first six attempts as shown in table 5.2 were not useful because of inaccurate scoring of features by the parsing method.

Table 5.18: The sequence of attempts for student number 20

Attempt No.	**Group Number**	**Attempt No.**	**Group Number**
1	2	4	2
2	2	5	2
3	2	6	6

Student number 20 attempted six times as shown in table 5.18. The student remained in the same state, which is student group two, during the first five attempts. The student transferred into student group six in the last attempt.

The initial levels of programming abilities for student number 20 were fair as shown in the first attempt. It includes a correct declaration of all required fields, a partial declaration of a constructor and all required set methods.

The analysis of the Java source codes for each student attempt showed that student number 20 improved his/her understanding of get methods and behaviour methods and programming abilities in writing get methods and behaviour methods. The gradual improvements of student number 20 during the course of the six attempts indicated the appropriateness or usefulness of the diagnostic feedback given at each attempt.

Table 5.19: The sequence of attempts for student number 23

Attempt No.	**Group Number**	**Attempt No.**	**Group Number**
1	3	5	4
2	3	6	4
3	3	7	4
4	3	8	7

Student number 23 attempted eight times. The student remained in the same

state, which is student group three, during the first four attempts. The student transferred into student group four in the fifth attempt and remained in that state for the next three attempts. Finally, the student group transferred into student group seven in the last attempt.

The initial levels of programming abilities for student number 23 were fair as shown in the first attempt. It includes a correct declaration of all required fields, a constructor with proper field initialisation and some of the required get methods. During the course of the eight attempts, the student improved his/her levels of programming abilities as they transferred into higher states, first into the student group four after fifth attempt and then into the student group seven in the last attempt.

The analysis of the Java source codes for each student attempt showed that student number 23 improved his/her understanding of set methods and behaviour methods and programming abilities in writing set methods and behaviour methods. The steady improvements of student number 23 indicated the appropriateness or usefulness of the diagnostic feedback given at each attempt.

Table 5.20: The sequence of attempts for student number 24

Attempt No.	**Group Number**
1	2
2	2
3	3
4	4

Student number 24 as shown in table 5.20 attempted four times. The student transferred into student group two for the first attempt and remained in that state in the second attempt. The student transferred into student group three in the third attempt and then transferred into student group four in the last attempt.

The initial levels of programming abilities for student number 24 were fair as shown in their first attempts. The first attempt of student number 24 includes a correct declaration of some of the required fields, incomplete declaration of a constructor and get methods. During the course of the four attempts, student number 24 improved his/her levels of programming abilities as he/she transferred into a higher state , which is student group three in third attempt and then into student group four in last attempt.

The analysis of the Java source codes for each student attempt showed that student number 24 improved his/her understanding of fields and constructors and programming abilities in writing constructors and set methods. These improvements indicated the positive impact of the diagnostic feedback given at each attempt.

5.5 State Transition Diagram

In the previous section, the sequences of attempts of each student were analysed using qualitative methodology. To assess the overall impact of the diagnostic feedback and reveal if there are any patterns on how students improve their object-oriented programming abilities, a state transition diagram was used that visualises the overall student interaction with the diagnostic feedback after each attempt. This technique had been also applied in chapter three.

The seven identified student group profiles were assigned a group number based on the levels of the programming abilities. The overall scores of the defined features based on the pattern of each student group profile were used to estimate the levels of programming abilities of each student group profile. Group one was

assigned to the lowest score whereas group seven was assigned to the highest score. Each student attempt is assigned an appropriate student group profile, which is treated as a state in the transition state diagram.

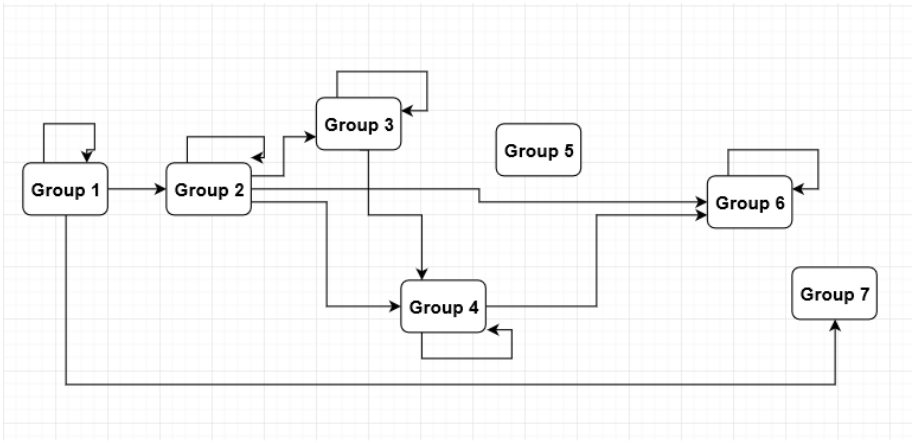


Figure 5.2: The state transition diagram of the overall sequence of student groups

As shown in figure 5.2, the overall impact of the diagnostic feedback was positive in improving the programming abilities of students as most transitions are forward from low levels of programming abilities to higher levels of programming abilities. The loop transitions shown in group 1, group 2, group 3, group 4, and group 6 indicated that some students could not transfer into a higher state after an attempt, instead they remained in the same state in the following one or more attempts.

There is some sort of pattern on how students improve their levels of object-oriented programming abilities. As the the state transition diagram reveals, most students don't necessarily need intermediate states to progress into higher states.

The state transition diagram revealed also the absence of any transition to or from group five. This indicated that there are no students who were assigned

to student group five. Based on the pattern of the student group five and the diagnostic feedback given to this student group, the difference between student group five and six is in the detail how the three main components of a class, which are fields, a constructor, and methods are implemented. It was also noticed in the previous section that the parsing method could not score correctly some of the defined features in some circumstances. These could be the possible reason why the student group five has no link in the state transition diagram.

5.6 Improvement of the Parsing Method

The comparison between automated and manual marking of the defined features conducted for all student attempts revealed that the proposed parsing method could not score correctly some of the defined features in some circumstances. In the following paragraphs, the proposed improvements of the parsing method are described.

The parsing method recognised only field declaration which ends with semi colon using the following pattern:

[Pp]ublic or [Pp]rivate or [Pp]rotected (data type) (field names) ;

The semi colon was added to the pattern to define field declaration to differentiate method declarations from field declarations. The pattern could be improved by allowing semi colon as an optional and ignoring any identified method declarations using a filter that detects any set of parameters enclosed within parenthesis. Once the field declarations are recognised correctly, identifying the correct data

type was not an issue for the automated marking of features related to correctness of fields except for the feature of field presence.

Constructors with void or any data type return types were not captured correctly using the current pattern, which only recognises declaration of constructors with the same name as the class name and no return type. The current pattern could be improved to include constructor declarations with return types using the following pattern:

[Pp]ublic or [Pp]rivate or [Pp]rotected (optional word for return type) class name (optional parameters) assignment statements;

The other issue with the current pattern was that class names with lower case were not recognised. This could be resolved by ignoring case sensitiveness between class name and constructor name.

The current pattern can only recognise set method declarations with void return type and only one assignment statement. Set methods with return types other than void were not identified using the current pattern. If set method declarations with return types other than void are included into the current pattern, they will be identified also as behaviour methods with only one assignment statement. To resolve this issue a filter should be added that checks the status of the assignment statement to make sure that one of the field names from the field declarations is on the left side of the assignment statement.

The limitation of the current pattern for capturing implemented behaviour methods is that it identifies only method implementations with specific names, which are specified when a programming exercise is set up. If a student chooses

a different name for a behaviour method that does not match one of the specified method names, the current pattern can not recognise it as the right behaviour method.

One of the solutions to improve the limitation of the current pattern that captures behavior methods is to capture all behaviour methods and mark against all possible method names and choose the best match. The other alternative is to specifically instruct students to name the behaviour methods according to recommended names as part of the description of a programming exercise.

5.7 Summary

The method proposed for identifying student group profiles based on student responses to an object-oriented programming exercise was evaluated based on data gathered from a trial. Before conducting a trial, first a diagnostic feedback was constructed for each identified student group. Secondly, a web-based formative assessment tool was designed and developed that integrates the constructed diagnostic feedback with the identified student group profiles.

To construct diagnostic feedback for the identified student group profiles, the learning outcomes described in the previous chapter and the patterns in each student group profile were analysed. The patterns in each student group profile is used to identify the current levels of object-oriented programming abilities and the learning outcomes is used to define the expected level of object-oriented programming abilities. Once the current levels and gaps of object-oriented programming abilities are identified for each student group profile, a diagnostic feedback is constructed that closes the gaps and improves the object-oriented programming

abilities.

A trial was conducted using the developed web-based formative assessment tool and thirty five students participated during the trial. The trial produced raw data and it was processed to make it suitable and improve its quality before conducting analysis and evaluation. The data preparation process generated three data sets. The first data set is a set of Java source codes for all attempts of twenty five student participants. The second data set is a sequence of attempts with their corresponding number, duration, assigned group number and Java file name for each of the twenty five student participants. The third data set is a score of each defined features for all student attempts using both automated and manual marking. The automated marking was done based on the proposed and developed parsing method.

Once the required data sets were prepared, the distribution of the number of attempts for the twenty five students was visualised using a bar chart. The bar chart indicated that 10 students attempted less than three times while 15 students attempted more than two times. The minimum and maximum number of attempts are 1 and 11 respectively. On average a student attempts 4.4 times.

Individual qualitative analysis was conducted for the 15 students who attempted more than two times. The purposes of the analysis were first to assess the effectiveness of the constructed diagnostic feedback in improving object-oriented programming abilities of students. Secondly, to measure the correctness of the proposed parsing method in capturing and scoring the defined features and how they affect the appropriateness and usefulness of a diagnostic feedback.

We applied the following analysis procedure for each of the selected student participant. First the sequence of attempts and their corresponding assigned

student group were analysed to describe the state or student group transitions of a student during the course of the assessment session. Secondly, the Java source code of all attempts of a student were analysed to identify the initial levels of programming abilities and to find out how the student progresses until the last attempt. Thirdly, the diagnostic feedback assigned to each student attempt were analysed to determine whether they were useful or appropriate. Finally, we analysed the comparison between the automated and manual marking of features for each student attempt to measure the correctness of the proposed parsing method in capturing and scoring features in order to assess their impact on the usefulness or appropriateness of the assigned student group or its corresponding diagnostic feedback.

From the individual qualitative analysis of the 15 student participant students, three categories of students were identified. The first group, which includes student number 2,8,9,15,and 19, showed slight improvements of levels of programming abilities due to the inaccurate capturing and scoring of features by the parsing method. The second group, which includes student number 11 and 13, did not show any improvements of levels of programming abilities due to poor previous knowledge and understanding of object-oriented programming. The final third group, which includes student number 4,5,7,10,16,20,23,and 24, showed gradual improvements of levels of programming abilities due to the usefulness and appropriateness of the diagnostic feedback they received during the assessment session.

To assess the overall impact of the diagnostic feedback and reveal if there are any patterns on how students improve their object-oriented programming abilities, a state transition diagram was used that visualises the overall student

interaction with the diagnostic feedback after each attempt. The state transition diagram showed that the overall impact of the diagnostic feedback was positive in improving the programming abilities of students as most transitions are forward from low levels of programming abilities to higher levels of programming abilities. The state transition diagram also revealed how students improve their levels of object-oriented programming abilities.

Finally, improvements for the proposed parsing method were suggested based on the analysis of the comparisons between automated and manual marking of the Java source codes of all attempts.

Chapter 6

Conclusion and Recommendations

6.1 Conclusion

The main aim of this research is to investigate the application of unsupervised learning, which is snap-drift modal learning neural network, in modelling student responses to multiple choice questions and object-oriented programming exercises. The purpose of student modelling is to identify student group profiles that support tutors in generating diagnostic feedback that can facilitate conceptual understanding of topics and development of basic object-oriented programming abilities. To achieve this aim, three research objectives were addressed.

Firstly, to conduct a literature review on the feedback mechanisms of currently existing computer-based formative assessments and application of machine learning in the context of virtual learning environments. Secondly, to improve the effectiveness of the previous application of snap-drift modal learning neural network in generating intelligent diagnostic feedback that can facilitate conceptual understanding of topics. Thirdly, to extend the application of snap-drift modal

learning neural network in generating intelligent diagnostic feedback that can facilitate development of basic object-oriented programming abilities.

A survey of existing computer-based formative assessments was conducted. The survey covered all types of assessment tasks such as multiple choice questions, short free text answers, and problem solving exercises. Secondly, the feedback mechanisms of all gathered computer-based formative assessments were analysed. The result showed that there are three types of feedback mechanisms. The other aspect of the literature review was to assess the application of machine learning in the context of virtual learning environments. Based on the result of both aspects of the literature review, it can be concluded that existing computer-based formative assessments have never utilised unsupervised machine learning to improve their feedback mechanisms. Machine learning techniques have been applied to construct student models, which are represented as categories of knowledge levels such as beginning, intermediate and advanced. The constructed student models don't specify what concepts are understood, the gap of understanding and misconceptions.

A method was proposed that improves the effectiveness of snap-drift modal learning neural network in identifying useful student group profiles and supporting tutors without the knowledge of machine learning in generating diagnostic feedback that improves conceptual understanding of students. The method was evaluated by conducting trials using two real assessment tasks. First student responses were gathered to identify student group profiles, which is the training stage of the snap-drift modal learning neural network. Once the training stage is finished, further trials were conducted to gather student responses for testing stage. Analysis of the gathered student responses from the two trials showed that

all of them were assigned to their appropriate student group profiles and the percentage of perfectly matched student responses could be improved by increasing the number of student group profiles and selecting student group profiles with no more than three most likely responses. The analysis of gathered student responses using state transition diagram also showed that the diagnostic feedback constructed based on the identified student group profiles has a positive impact on improving the learning performance of students.

During the second phase of the research, the proposed novel method was extended in order to identify useful student group profiles that represent different programming abilities of writing an object-oriented class. The purpose of identifying student group profiles was to facilitate construction of diagnostic feedback that improves the development of basic object-oriented programming abilities. The extended method was also evaluated by conducting trials to gather student responses to an object-oriented programming exercise using individual qualitative analysis and state transition diagram techniques.

The result of the individual qualitative analysis of 15 students, who attempted more than two times, revealed three categories of students. The first group showed slight improvements of levels of programming abilities due to the inaccurate capturing and scoring of features by the parsing method. The second group did not show any improvements of levels of programming abilities due to poor previous knowledge and understanding of object-oriented programming. The final third group showed gradual improvements of levels of programming abilities due to the usefulness and appropriateness of the diagnostic feedback they received during the assessment session.

The state transition diagram showed that the overall impact of the diagnos-

tic feedback was positive in improving the programming abilities of students as most transitions are forward from low levels of programming abilities to higher levels of programming abilities. Even though, no clear pattern on how students improve their levels of object-oriented programming abilities was revealed from the state transition diagram, it obviously revealed that students do not necessarily need intermediate states to progress into higher states. The state transition diagram revealed also the absence of any transition to or from group five. Finally, improvements for the proposed parsing method were suggested based on the analysis of the comparisons between automated and manual marking of the Java source codes of all attempts.

6.2 Knowledge Contribution and Limitations

Overall the main objectives of this research project were addressed successfully. In addition to enhancing the insights gained into an application of snap-drift modal learning neural network in new application domains, it was also demonstrated how existing web-based formative assessments could be improved by integrating identified student group profiles and constructed diagnostic feedback to enhance conceptual understanding of topics and development of basic-object oriented programming abilities. These improved web-based formative assessments will have a significant impact on improving student learning experience and supporting tutors. Tutors will not need the knowledge of unsupervised machine learning technique to identify student group profiles that facilitate generation of diagnostic feedback. In addition to this, tutors can understand how students learn topics or develop basic object-oriented programming abilities by visualising the sequence

of student attempts and their assigned student group using state transition diagram.

There were limitations of the research project, which could improve the result of the research. Since training data are the main source of knowledge for unsupervised learning techniques in general and snap-drift modal learning neural network in particular, increasing the size and coverage of gathered student responses would have enhanced the outcome of the research. The available training data sets were limited as the research project considered only students enrolled at London Metropolitan University. The number of trials conducted were also limited due to the fact that two trials have to be conducted at different academic years so that different cohort of students can be participated during training and testing. Finally, the accuracy of the training patterns gathered from object-oriented programming exercises were influenced by the capability of the parsing method, which applied a rule based technique.

6.3 Recommendations for Future Research

Based on the conclusion, the following are recommended for future research:

1. To explore alternative representation techniques that minimise the dimension of training patterns. Representing training patterns gathered from programming exercises as set of binary vectors is not efficient as it increases the dimension of the training pattern. For example, the dimension of training patterns gathered from an object-oriented programming exercise represented by 24 defined features is around 80.

2. To extend the defined features that represent student responses to an object-oriented programming exercise in order to enable it to represent student responses to advanced object-oriented programming exercises that assess student's programming abilities in writing complex classes using advanced object-oriented programming concepts such as inheritance and polymorphism.

3. To investigate if it is possible to capture features from student responses with out the use of parsing methods.

4. To conduct more trials in order to evaluate the effectiveness of the proposed methods for topics from different subject areas and different object-oriented programming languages such as C++ or C#.

5. To improve and evaluate the developed web-based formative assessment.

6. To extend the research on applying snap-drift modal learning neural network to other forms of assessment tasks such as free-text response, problem solving and essay writing.

Appendix A

A set of five multiple choice questions regarding probability concept

A sample of adults in a particular profession was surveyed and classified by gender and occupational status as follows:

Gender	Full-time work	Part-time work	Unemployed
Male	267	134	126
Female	75	78	97

Answer the following five questions based on the above table.

1. Estimate the probability that a randomly sampled adult will be unemployed.

- 28%
- 0.29
- 0.24
- 14%

2. Estimate the probability that a randomly sampled adult will be a male in full-time work.

- 0.78
- 50%
- 78%
- 0.44
- 0.34

3. Estimate the probability that a randomly sampled adult will be in part-time work given he is male.

- 0.25
- 0.17
- 63.2%
- 0.63
- 25.4%

4. Estimate the probability that a randomly sampled adult will be in part-time work given she is female.

- 0.37
- 0.10
- 37%
- 31.2%
- 0.31

5. Which of the following statement is false ?

- The employment status and gender are independent
- At least 75% of the men have some employment
- The ratio of men to women in the profession is approximately 2:1
- Gender appears to affect employment status

Appendix B

A set of five multiple choice questions regarding a concept from object oriented programming

1. Why does a class need to have a constructor ?

- to return the values of its fields
- to make an instance of the class
- to declare an object
- to change the values of its fields

2. What does a constructor return ?

- String
- void
- It does not return anything
- int

3. Which of the following is not correct about a constructor

- It is a method that returns the values of fields
- The name of a constructor and its class are always the same
- It can be used to initialise the values of fields when a new object is created
- It is a method that is executed when a new object is created

4. Consider a class named Circle. Which of the following is the correct header for a constructor of Circle class with no parameters ?

- public void Circle()
- public Circle()
- public class Circle()
- public String Circle()

5. Consider a class named Book, which has three fields: author, title and pages.
Its constructor initialises the author and title to user input values and sets the value of pages to its default value of 0.
Which of the following is the correct way to create an object fo type Book ?

- x = new Book(author, title, pages);
- x = new Book(author, title, 0);
- x = new Book(author, title);
- x = new Book();

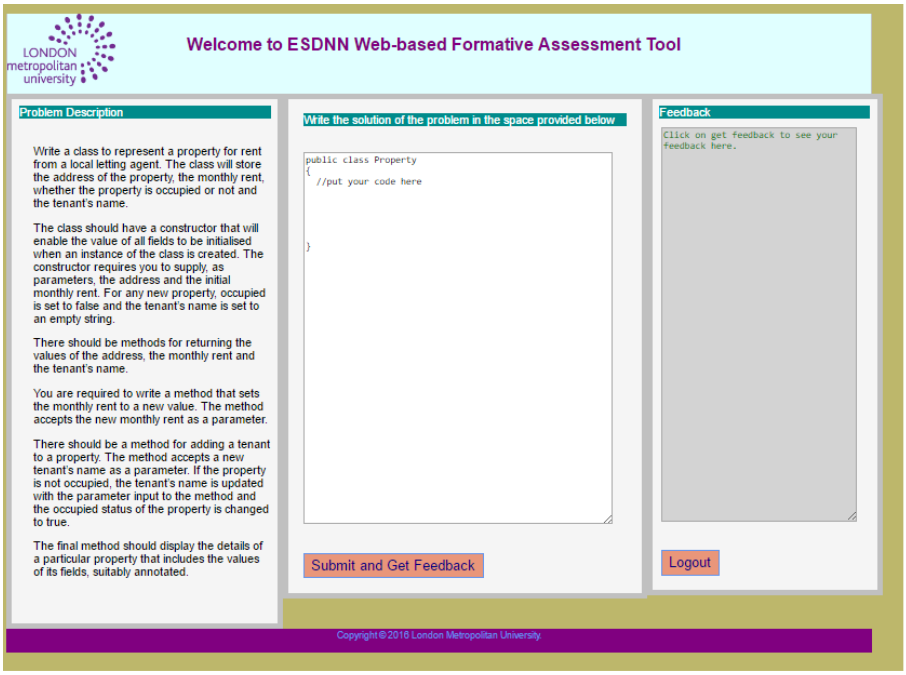
Appendix C

A model answer to an object-oriented programming exercise

```
public class Property {
	private String address;
	private int monthlyRent;
	private String tenantName;
	private boolean isOccupied;
	public Property (String newAddress, int newInitialMonthlyRent)	{
		address = newAddress;
		monthlyRent = newInitialMonthlyRent;
		tenantName = "";
		isOccupied= false;
	}
	// Get methods.
	public String getAddress () {
		return address;
	}
	public String getTenantName () {
		return tenantName;
	}
	public int getMonthlyRent () {
		return monthlyRent;
	}
	// Set methods
	public void setMonthlyRent (int newInitialMonthlyRent){
		monthlyRent = newInitialMonthlyRent;
	}
	// Behaviour monthlyRent
	public void addNewTenant (String newTenantName) {
		if (!isOccupied) {
			tenantName = newTenantName;
			isOccupied = true;
		}
	}
	public void display () {
		System.out.println("Tenant name: " + tenantName);
		System.out.println("Address: " + address);
		System.out.println("Monthly rent: " + monthlyRent);
		System.out.println("Ocupied status: " + isOccupied);
	}
}
```

Appendix D

A screen shot of a web-based formative assessment



References

[1] The concepts of feedback and feed forward. https://www.jisc.ac.uk/guides/feedback-and-feed-forward. [Online; accessed March 23, 2017]. 30

[2] The Seven Learning Styles. https://www.learning-styles-online.com/overview/. [Online; accessed March 20, 2017]. 27

[3] A.Eckerdal and M. Thune. Novice java programmers conceptions of object and class, and variation theory. In *2005 10th Annual Conference on Innovation and Technology in Computer Science Education*, Monte de Caparica, Portugal, 2005. 54

[4] J.L.F. Aleman, D. Palmer-Brown, and C. Jayne. Effects of response-driven feedback in computer science learning. *Education, IEEE Transactions on*, **54**[3]:501–508, Aug 2011. 2, 69

[5] S. Amershi and C. Conati. Unsupervised and supervised machine learning in user modeling for intelligent learning environments. In *IUI '07: Proceedings of the 12th international conference on Intelligent user interfaces*, pages 72–81, New York, NY, USA, 2007. ACM. 6

[6] D. Barnes and M. Kolling. *Objects First with Java*. Pearson Education Ltd, UK, 2009. 93, 103

[7] B. Bell and B. Cowie. The characteristics of formative assessment in science education. *Science Education*, **85**:536–553, 2001. 30, 68

[8] S. Benford, E. Burke, E. Foxley, and C. Higgins. The celidh system for the automatic grading of students on programming courses. In *Proceedings of the 33 Annual ACM Southeast Conference*, 1995. 47

[9] S. Bennett, S. McRobb, and R. Farmer. *Object-Oriented Systems Analysis and Design*. McGraw-Hill Higher Education, UK, 2010. 92

[10] P. Black and D. William. Assessment and classroom learning. *Assessment in Education:Principles,Policy and Practice*, **5**[1]:7–68, 1998. 30, 69

[11] P. Blayney and M. Freeman. Individualised interactive formative assessments to promote independent learning. *Journal of Accounting Education*, **26**:155–165, 2008. 2, 43

[12] J. D. Bransford, A. L. Brown, and R. T. Cocking. *How people learn: Mind, brain,experience and school*, pages 8–27. National Academy Press, expanded ed. edition, 2000. 29

[13] I. Bruno and L. Santos. Written comments as a form of feedback. *Studies in Educational Evaluation*, **36**:111–120, 2010. 30, 69

[14] T. Buchanan. The efficacy of world-wide web mediated formative assessment. *Journal of Computer Assisted Learning*, **16**:193–200, 2000. 29, 31

[15] J. A. Bullinaria. Introduction to neural computation. http://www.cs.bham.ac.uk/~jxb/inc.html, 2012. [On-line; accessed February 11, 2012]. 9, 11, 12, 13, 14

[16] C. Carmona, G. Castillo, and E. Millan. Designing a dynamic bayesian network for modeling students' learning styles. In *Eighth IEEE International Conference on Advanced Learning Technologies*, pages 346–350, Santander, Cantabria, Spain, 2008. 28

[17] J. Carneson, G. Delpierre, and K. Masters. Designing and managing multiple choice questions. Technical report, 2002. 4, 33, 53

[18] K. M. Cauley and J. H. McMillan. Formative assessment techniques to support student motivation and achievement. Technical report, Virginia Commonwealth University, 2010. 29

[19] S. Y. Chen and X. Liu. An integrated approach for modeling learning patterns of students in web-based instruction: A cognitive style perspective. *ACM Transactions on Computer-Human Interaction*, **15**[1], 2008. 29

[20] T. Chumwatana, K. W. Wong, and H. Xie. A som-based document clustering using frequent max subtrings for non-segmented texts. *Intelligent Learning Systems Applications*, **2**:117–125, 2010. 19

[21] C. Daly, N. Pachler, Y. Mor, and H. Mellar. Exploring formative e-assessment: using case stories and design patterns. *Assessment and Evaluation in Higher Education*, **35**[5]:619–636, 2010. 31, 32

[22] Rick Decker and Stuart Hirshfield. Top-down teaching: object-oriented programming in cs 1. In *ACM SIGCSE Bulletin*, **25**, pages 270–273. ACM, 1993. 92, 93

[23] M. Dittenbach, A. Rauber, and D. Merkl. Uncovering hierarchical structure in data using the growing hierachical self-organizing map. *Neurocomputing*, **48**:199–216, 2002. 19

[24] H. Donelan, C. Pattinson, D. Palmer-Brown, and S. W. Lee. The analysis of network manager's behaviour using a self-organising neural networks. *Proceedings of The 18th European Simulations Multiconference*, pages 111 – 116, 2004. 24

[25] E. Frias-Martinez, G. Magoulas, S. Chen, and R. Macredie. Modelling human behaviour in user-adaptive systems: Recent advances using soft computing techniques. *Expert Systems with Applications*, **29**:320 – 329, 2005. 6, 55

[26] X. Fu, B. Peltsverger, and J. Liu. Apogee-automated project grading and instant feedback system for web based computing. In *39th SIGCSE technical symposium on Computer Science Education*, Portland, USA, 2008. ACM. 2, 44

[27] D. Gilmore and J. Self. *Artificial Intelligence and Human Learning: Intelligent Computer-Aided Learning*. Chapman and Hall, London, UK, 1988. 25

[28] C. Gonzalez, J. C. Burguillo, and M. Llamas. A qualitative comparison of techniques for student modeling in intelligent tutoring systems.

In *Frontiers in Education, 36th Annual Conference*, pages 13–18, San Diego, CA, USA, 2006. 25

[29] S. Haykin. *Neural Networks and Learning Machines*. Pearson Education, New Jersey, USA, 2009. 6, 9, 10, 11, 15, 16

[30] M. Homsi, R. Lutfi, C. R. Maria, and G. Barakat. Student modeling using nn-hmm for efl course. In *2008 3rd International Conference on Information and Communication Technologies: From Theory to Applications*, pages 1–6, Damascus, Syria, 2008. 1, 26

[31] Hotpotatoes. Hot potatoes web based assessment tool. https://www.questionmark.com/us/Pages/default.aspx, 2012. [On-line; accessed on November 30, 2012]. 2, 35

[32] Nan-Chen Hsieh. An integrated data mining and behavioral scoring model for analyzing bank customers. *Expert Systems with Applications*, **27**:623–633, 2004. 19

[33] Jisc. Short answer marking engines. http://www.jisc.ac.uk/media/documents/projects/shorttext.pdf. [Online; accessed Feburary 12, 2013]. 37

[34] S. Jordan. Short-asnwer e-assessment questions: five years on, 2012. 37

[35] S. Jordan and T. Mitchell. e-assessment for learning? the potential of short-answer free-text questions with tailored feedback. *British Journal of Educational Technology*, **40**[2]:371–385, 2009. viii, 2, 37, 39, 41

[36] M. Kang and D. Palmer-Brown. A modal learning adaptive function neural network applied to hand-written digit recognition. *Information Sciences*, **178**[20]:3802–3812, 2008. 21

[37] S. Kaski, T. Honkela, K. Lagus, and T. Kohonen. Websom-self organizing maps of document collections. *Information Sciences*, **163**:135–156, 2004. 19

[38] T. Kohonen, M. R. Schroeder, and T. S. Huang. *Self-Organizing Maps*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 3rd edition, 2001. viii, 15, 16, 17, 19, 20, 21

[39] Michael Kolling and John Rosenberg. Guidelines for teaching object orientation with java. In *Proceedings of the 6th conference on Information Technology in Computer Science Education (ITiCSE)*. ACM, 2001. 92, 93

[40] C. Koutsojannis and A. Hatzilgeroudis. Web based intelligent tutoring system teaching nursing students fundamental aspects of biomedical technology. In *the 23rd annual EMBS international conference*, Estambul, 2001. 24

[41] K. Lagus, S. Kaski, and T. Kohonen. Mining massive document collections by the websom method. *Information Sciences*, **163**:135–156, 2004. 19

[42] S. W. Lee and D. Palmer-Brown. Phonetic feature discovery in speech using snap-drift. In *International Conference on Artificial Neural Networks*, LNCS, pages 952–962, 2006. 21, 24

[43] C. Li and J. Yoo. Modeling student online learning using clustering. In *ACM-SE 44: Proceedings of the 44th annual Southeast regional conference*, pages 186–191, New York, NY, USA, 2006. ACM. 28

[44] H. Liu, H. Shi, and Y. Shang. Student modeling with timed assessment information. In *ITRE 2004 2nd International Conference Information Technology: Research and Education*, pages 121–125, London, England, UK, 2004. 1, 26

[45] K. L. McGraw and K. Harbison-Briggs. *Knowledge Acquisition: Principles and Guidelines*. Prentice-Hall Inc., USA, 1989. 6

[46] D. Merkl. Text classification with self-organizing maps: some lessons learned. *Neurocomputing*, **21**:61–77, 1998. 19

[47] A. H. Miller, B. W. Imrie, and K. Cox. *Student Assessment in Higher Education*. Kogan Page Limited, UK, 1998. 33

[48] C. Murphy, G. Kaiser, K. Loveland, and S. Hasan. Retina: Helping students and instructors based on observed programming activities. In *SIGCSE'09*, Tennessee, USA, 2009. 2, 40

[49] D. Nicol. E-assessment by design:using multiple-choice tests to good effect. *Journal of Further and Higher Education*, **31**[1]:53–64, 2007. 33, 53

[50] Y. Nouh, P. Karthikeyani, and R. Nadarajan. Intelligent tutoring system-bayesian student model. In *2006 1st International Conference on Digital Information Management*, pages 257–262, 2007. 24

[51] E. Oja. Unsupervised learning in neural computation. *Theoretical Computer Science*, pages 187–207, 2002. 14

[52] N. Pachler, C. Daly, Y. Mor, and H. Mellar. Formative e-assessment: Practitioner cases. *Computers and Educations*, **54**:715–721, 2009. 32

[53] D. Palmer-Brown, S. W. Lee, C. Jayne, and M. Kang. Modal learning neural networks. *WSEAS Transactions on Computers*, **8**[2]:222 – 236, 2009. 2, 21, 59, 69

[54] QAA. Code of practice for the assurance of academic quality and standards in higher education-section 6: Assessment of students. http://www.qaa.ac.uk/Publications/InformationAndGuidance/Documents/, 2010. [On-line; accessed on September 15, 2010]. 29

[55] Questionmark. Questionmark perception web based assessment tool. https://www.questionmark.com/us/Pages/default.aspx, 2012. [On-line; accessed on October 12, 2012]. viii, 2, 35, 36, 50

[56] Richard J. Riding and Eugene Sadler-Smith. Cognitive style and learning strategies: Some implications for training design. *International Journal of Training and Development*, **1**[3]:199–208, 1997. 28

[57] J. Sandberk and J. Andriessen. Where is ai and how about education. *Artificial Intelligence in Education*, pages 545–552, 1997. 2

[58] V. J. Shute. Focus on formative feedback. *Review of Educational Research*, **78**:153–189, 2008. 30, 68, 69

[59] R. Stathacopoulou, G. D. Magoulas, and M. Grigoriadou. Neural network-based fuzzy modeling of the student in intelligent tutoring systems. In *IJCNN'99 International Joint Conference on Neural NetworksIntelligent Systems Design and Applications (ISDA)*, pages 3517–3521, Washington, DC, USA, 1999. 1, 27

[60] K.L Stewart and L.A Felicetti. Learning styles of marketing majors. *Educational Research Quarterly*, **15**[2]:15–23, 1992. 27

[61] J. Z. Sukkarieh and J. Blackmore. c-rater: Automatic scoring for short constructed responses. In *Proceedings of the Twenty-Second International FLAIRS Conference*, 2009. 2, 37

[62] J. Z. Sukkarieh and S. G. Pulman. Information extraction and machine learning: Auto-marking short free text responses to science questions. *Artificial Intelligence in Education*, **125**:629–637, 2005. 2, 37, 38

[63] P. Symeonidis. *Automated Assessment of Java Programming Coursework for Computer Science Education*. PhD thesis, University of Nottingham, 2006. viii, 47, 48, 49

[64] N. Truong, P. Bancroft, and P. Roa. Learning to program through the web. In *10th annual SIGCSE conference on Innovation and technology in Computer Science Education*, pages 9–13, Monte de Caparica, Portugal, 2005. ACM. 2, 45

[65] UCI. Uci machine learning repository:iris data. https://archive.ics.uci.edu/ml/datasets/Iris, 2012. [On-line; accessed on February 21, 2012]. 57

[66] R. Walker, J. Voce, and J. Ahmed. 2012 survey of technology enhanced learning for higher education in the uk. Technical report, Ucisa, 2012. 1

[67] T. Wang. Web-based quiz-game-like formative assessment: Development and evaluation. *Computers and Educations*, **51**:1247–1263, 2008. 2, 34, 35

[68] B. Yusob, S. Mariyam H. Shamsuddin, and N. B. Ahmad. Developing student model using kohonen network in adaptive hypermedia learning system. In *2009 Ninth International Conference on Intelligent Systems Design and Applications (ISDA)*, pages 938–943, Pisa, Italy, 2009. 1, 26

[69] . Effective practice with e-assessment: An overview of technologies, policies and practice in further and higher education. Technical report, HEFCE, 2007. 31