LSTM Cluster: An Integrated Approach to Cluster Students' Problem Solving Sequences in Log Files

by

Qi Guo

A thesis submitted in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

in

Measurement, Evaluation and Cognition

Department of Educational Psychology

University of Alberta

Abstract

Modern technology-based assessments have the capacity to record every student-computer interaction in log files. Cluster analysis of log files could yield insights about students' problem solving strategies and their misconceptions. However, current cluster analysis algorithms often rely on expert-selected features and they do not take the order of student actions into account. To address these limitations, this study proposes a novel deep learning approach to cluster student actions in log files. The proposed method, *Long Short Term Memory (LSTM) cluster*, first extracts features relevant to the problem from sequential data, and then clusters students based on the extracted features. In order to demonstrate and evaluate LSTM cluster, a real data study and a simulation study were conducted. In the real data study, LSTM cluster identified four different problem-solving strategies. A detailed examination of these strategies suggested that the order of student actions was indeed important. The simulation study suggested that LSTM cluster had good accuracy for large sample sizes. However, when sample size is small, the number of student actions needs to be reduced in order to prevent overfitting.

AKNOWLEDGEMENTS

My PhD journey is a long one. There are so many people I need to acknowledge. First, I want to express gratitude to my supervisor, Dr. Ying Cui, for guiding me throughout my journey. I made many mistakes during the process, but she is always patient and kind with me. Next, I want to thank the rest of the CRAME professors whom I have worked so closely during my PhD years. Dr. Jacqueline Leighton taught me how to write academic articles with clarity, and how to explore new areas of research despite all the challenges. Dr. Mark Gierl showed me the big picture and future direction of the field. He made me realize the importance of data science and how to make research meaningful by aligning with industries. Dr. Okan Bulut worked so close with me. His door is always open. Whenever I have any technical questions, he is there to help. Dr. Maria Cutumisu is so supportive and friendly. She introduced me to many computer science professors, and helped me with several exciting research projects. I am so grateful I studied in CRAME because all the professors are so supportive and knowledgeable.

I will move on to thank the CRAME students. They are my dearest friends. In the beginning of my graduate study, Alex Riedel showed me what intrinsic motivation is. He learns for pure curiosity and the desire to help others. I adopt his motivations for my career and for my life. Then, I need to thank Wei Tang who studied together with me during the most difficult time of my PhD journey. Without her, I would not be able to master the mathematics that is required for my dissertation. I also want to thank the rest of the CRAME students who made my PhD journey so enjoyable and meaningful.

In the end, I want to thank my parents for all their unconditional love and support.

Contents

Abstract ii

AKNOWLEDGEMENTS iii

List of Tables vii

List of Figures viii

Chapter 1 Introduction 1

Organization of the Dissertation 4

Chapter 2 Literature Review 5

Technology-Based Assessments 5

Clustering Techniques in Educational Data Mining 13

Hard versus soft clustering techniques 14

Partitional versus hierarchical clustering techniques 14

Probabilistic versus heuristic clustering techniques 14

K-means clustering 15

Finite mixture models 16

Agglomerative cluster analysis 18

Determining the number of clusters 20

Applications of cluster analysis in educational data mining 25

Classification in Educational Data Mining 28

Deep learning 31

Deep Feedforward Neural Networks 32

Long Short Term Memory Model 36

Applications of deep learning in educational data mining 42

Sequential Pattern Analysis in Educational Data Mining 43

An introduction to sequential pattern analysis 44

Three steps of applying sequential pattern analysis 45

Applications of sequential pattern analysis in educational data mining 46

Research Problem 47

Chapter 3 Proposed Method: LSTM Cluster 49

LSTM Classification 49

Cluster Analysis 53

Interpretation of the Clusters 54

Chapter 4 Analysis of Real log files with the LSTM Cluster 56

Method 56

Participants 56

Description of the TRE science laboratory problem 57

Analysis 60

Results and Discussion 63

Review of the log files 63

LSTM cluster 65

Chapter 5 A Simulation Study 72

Method 72

Graph strategy 73

Table strategy 75

Try-them-all strategy 75

Random try strategy 76

Guess strategy 76

Analysis 76

Evaluation criteria 77

Results 79

Chapter 6 Discussion 83

Implications 83

Limitations and Recommendations for Future Research 84

Significance 85

REFERENCES 87

List of Tables

Table 1. *Lance* and *Williams* (1967) *Formula Coefficients for Different Cluster Distances*. 19

Table 2. *Actions from TRE simulation log files.* 59

Table 3. *Cluster descriptive statistics*. 67

Table 4. *Exemplar sequences and sequential patterns for each identified cluster*. 69

Table 5. *Mean prediction and cluster error rates for all simulation conditions.* 81

List of Figures

Figure 1. A screenshot of *Mysterious Plants*..... 7

Figure 2. A screenshot of *Save Patch*..... 8

Figure 3. Example Agglomerative Cluster Dendrogram of five cases. Reprinted from *Cluster Analysis* (5th Edition), by Everitt, B.S., (2011), Chichester, West Sussex, U.K: Wiley. 20

Figure 4. The conceptual diagram of a feedforward neural network. 33

Figure 5. Two ways to represent a recurrent neural network: (a) a folded representation of a recurrent neural network, and (b) an unfolded over time representation of (a). 37

Figure 6. A conceptual diagram for a LSTM network unfolded over time..... 39

Figure 7. A conceptual diagram for a LSTM network unfolded over time. X = input, Y = output, Sig.=sigmoid function, Tanh = hyperbolic tangent function, Out. gate = output gate, In. gate = input gate, F. gate = forget gate, C = cell, T = sequence size. 52

Figure 8. Scatterplot of simulated dataset with five clusters nested in three bigger clusters. 53

Figure 9. A snapshot of the TRE science laboratory. 59

Figure 10. LSTM output unit probability histogram for real data..... 66

Figure 11. Examples of LSTM output unit probability histograms for all six simulation conditions. 82

Chapter 1 Introduction

The widespread use of advanced computer technologies has begun to revolutionize educational assessments. Starting from early elementary grades, newly developed educational assessments become more and more technology-based, game-like, and presenting real world problems in a more authentic way (e.g., Escueta, Quan, Nickow, & Oreopoulos, 2017). Compared with traditional assessments, new technology-based assessments have the advantages of being more engaging, and allowing students to explore, experiment, get automated feedback, and interact with other students within a virtual environment. In addition, most of these technology-based assessments can store every action a student takes into a *log file*, which provides rich information about the student's thoughts and problem solving processes without interrupting the student's problem-solving process. As a result, technology-based assessments are promising for assessing process-based, higher-order thinking skills that are crucial for students to succeed in the 21st century. More specifically, the analyses of student log files have the potential to help answer the questions of how students solve problems, what strategies students use to solve problems, what common erroneous rules and/or misconceptions students have. While the log file provides comprehensive information in understanding students' problem solving processes, its complexity also poses a number of challenges for analysis (García, Romero, Ventura, de Castro, & Calders, 2011).

Kerr (2015) summarized the literature and pointed out three major challenges for analyzing data from log files (i.e., log data). First, log files contain huge quantities of information. For example, each participant can generate thousands of pieces of information in just half an hour of game play. Second, log files contain highly detailed information (e.g., "select Tool A,"

"move game character to the left," etc.), and there is no initial theory to indicate which actions are relevant to the problem and which are not (Levy, 2012; also, Leighton & Chu, 2016). Third, there is often little overlap between students' log files. In other words, log data are very sparse. Most of the information may not be directly relevant to the problem. In addition to these three challenges, it is also important to note that log file information is sequential in nature. That is, doing action A and then doing action B may be different from doing action B and then doing action A, even though the same actions are performed. The orders in which students perform certain actions may be crucial to understand their problem-solving strategies.

Traditional statistical methods designed for well-defined problems cannot be directly applied for the analysis of log data. Therefore, various data mining techniques have been applied to analyze log files. The detailed data mining techniques and examples of applications are summarized in Romero, Ventura, Pechenizkiy, and Baker's (2011) *Handbook of Educational Data Mining*, which is discussed in the literature review section of this dissertation. There are three major categories of data mining methods: classification methods such as Bayesian Network, and Neural Network (Hämäläinen, & Vinni, 2011), clustering methods such as k-means, self-organizing map (Vellido, Castro, & Nebot, 2011), and sequential pattern analysis (Zhou, Xu, Nesbit, & Winne, 2011). Among these three categories, classification methods and clustering methods often ignore students' orders of actions because they often use only action frequency as input (Hämäläinen, & Vinni, 2011; Vellido et al., 2011). As a result, potentially important information regarding student problem solving sequences of student actions might be overlooked. While sequential pattern analysis considers the orders of student actions, it often fails to distinguish problem-relevant actions and problem-irrelevant actions (Zhou et al., 2011).

Consequently, sequential pattern analysis tends to produce too many sequential patterns that are difficult to interpret substantively for meaning.

In order to address the challenges of log file analysis and the limitations of current educational data mining methods, the goal of this study was to propose and evaluate an integrated three-step analysis approach for log data analysis. In step 1, I proposed to use a deep learning neural network, *Long Short Term Memory Model* (LSTM), to predict the problem solving outcomes based on students' problem solving sequences. Deep learning neural networks are currently one of the most powerful machine learning algorithms for finding patterns in large data sets (Hinton et al., 2012). These networks work by extracting multiple layers of non-linear features from the raw input and using these features to predict the outcomes. The LSTM model, a specific type of deep learning neural network, can use sequential data to predict outcomes, and it has the ability to identify and ignore outcome-irrelevant input (Gers, Schmidhuber, & Cummins, 2002). These properties make the LSTM a promising approach to analyzing log files. The purpose of analysis at step 1 is to extract problem relevant features from the problem-solving sequences. In step 2, cluster analysis is used to group the results (i.e., the extracted features) from the LSTM network. In this step, students who use similar problem-solving sequences/strategies are clustered together. In step 3, sequential pattern analysis is used to identify and interpret each cluster's common sequences. The goal of step 3 is to identify what common problem solving strategy students in each cluster use. In addition, exemplar sequences that are close to the centroid of each cluster are identified to help interpretation. Therefore, the proposed method was named *LSTM cluster*.

In order to evaluate LSTM cluster, a real data study and a simulation study were conducted. The specific research questions are presented at the end of Chapter 3.

Only very recently have deep learning neural networks been applied in the field of educational data mining (Piech et al., 2015; Huang & Brusilovsky, 2016). This study is an attempt to bring powerful machine learning technology into educational data mining for the analysis of log data. It provides a way to integrate several current educational data mining methods in the analysis of log files to facilitate the classification of students' problem solving strategies. The results from the analysis of log files can help identify students' weaknesses in problem solving and have the potential to assist teachers in designing effective learning interventions.

Organization of the Dissertation

Chapter 1 contains the introduction of the dissertation. Chapter 2 reviews the literature of current educational data mining methodology in log file analysis. Chapter 3 presents the proposed LSTM cluster method. Chapter 4 presents the method and results for the real data study. Chapter 5 presents the method and results for the simulation study. In chapter 6, the implications, limitations, and future directions are discussed.

Chapter 2 Literature Review

This chapter begins by introducing technology-based assessments in general and reviewing game-based and simulation-based assessments in particular. Next, three major categories of techniques for analyzing log data in educational data mining are reviewed: clustering, classification, and sequential pattern analysis. In the end, the limitations of the current methods for analyzing log data are summarized.

Technology-Based Assessments

The field of *Technology-Based Assessment* is relatively new, and yet it is very diverse as it includes many different assessment formats, such as game-based assessments, simulation-based assessments, group assessments, and automated diagnostic assessments (Mayrath, Clarke-Midura, Robinson, & Schraw, 2011). In general, it reflects educational researchers' attempts to use modern technology to break the limitations of traditional assessments (e.g., costly performance based assessments, difficulty to capture student problem solving processes, & less engaging). Despite the increasing diversity in technology-based assessments, a majority of modern technology-based assessments are computer-based assessments that have the potential to assess student problem solving processes via computer generated log files. As explained before, a log file is the history of everything a student does while solving tasks during a technology-based assessment. Log files allow the capture of a rich amount of information about students' problem-solving processes without any interruption, which is one major breakthrough that separates modern technology-based assessments from traditional assessments. Thus, in this dissertation, I am particularly interested in technology-based assessments that include log files to assess students' learning/problem solving processes. Among different types of technology-based assessments, game-based and simulation-based assessments are the most commonly used

methods to capture students' detailed problem solving processes. Since the goal of the dissertation is to develop a method to cluster problem-solving sequences, simulation-based and game-based assessments are the focus in the following subsections.

In game-based and simulation-based assessments, students need to apply their knowledge and skills to dynamically interact with computers in virtual environments to solve problems (Behrens, Mislevy, DiCerbo, & Levy, 2011; Quellmalz et al., 2011; Zapata-Rivera, & Bauer, 2011). The major difference between game-based and simulation-based assessments is that simulation-based assessments involve realistic scenarios to frame the problems while game-based assessments involve imaginative and engaging but not necessarily realistic scenarios to frame the problems (Quellmalz et al., 2011; Zapata-Rivera, & Bauer, 2011). In other words, simulation-based assessments are designed to test whether students can solve problems in realistic situations, while game-based assessments are designed to be more entertaining and motivate student to learn and solve problems while playing.

In order to illustrate the two types of assessments, some examples are provided below. The *Mystery Plants* interactive computer task created by *National Assessment of Educational Progress* (NAEP; National Assessment of Educational Progress, 2009) is an example of a simulation-based assessment. An example screen shot is provided in Figure 1. In this particular task, students are asked to solve the problem of how much sunlight different plants need to grow well. In order to solve the problem, students need to conduct a series of virtual experiments by dragging different computer-simulated plants into different levels of a computer-simulated greenhouse and observe the plants' growth. Students need to make hypotheses before conducting the experiments and keep revising their hypotheses until they think they have enough evidence

(i.e., experiment results in tables and figures) to support their hypotheses. Since this task requires students to apply their scientific inquiry skills and to dynamically interact with a virtual scenario that resembles a real scientific experiment, this is considered a simulation-based assessment.

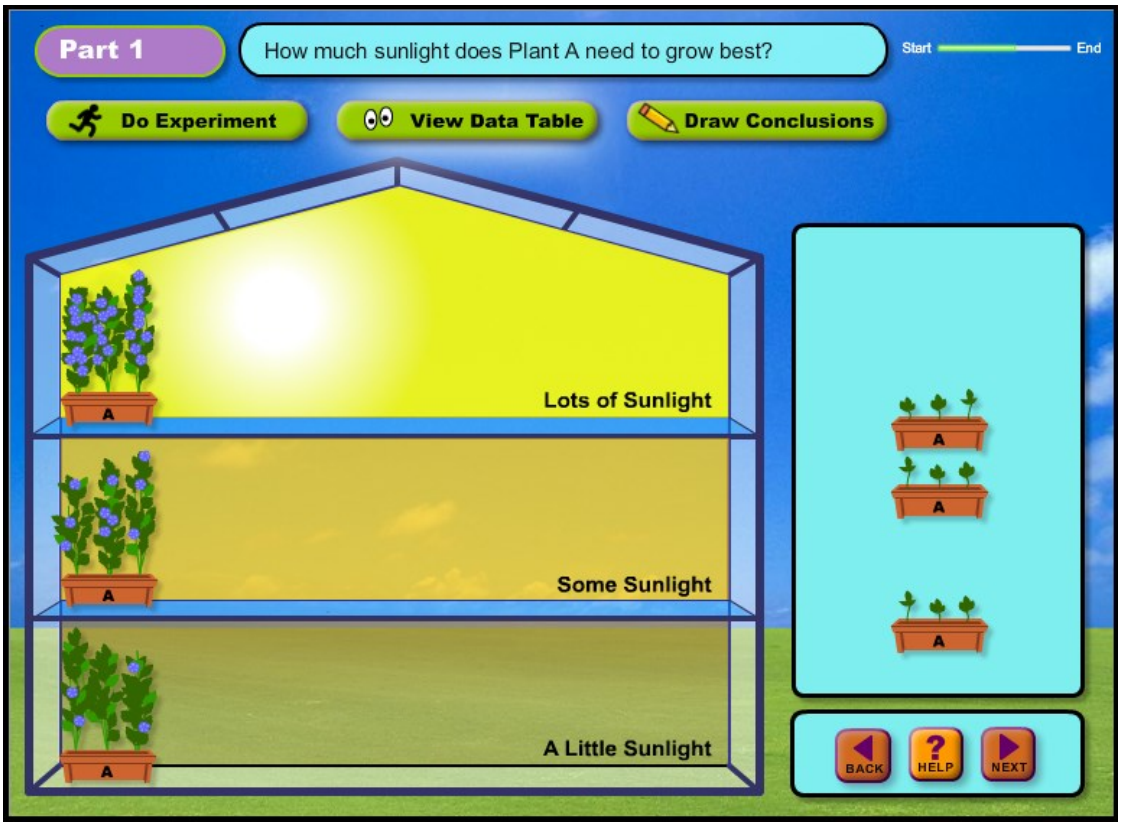

Figure 1. A screenshot of *Mysterious Plants*.

An example of game-based assessment is the learning game, *Save Patch*, designed by Baker, Chung, and Delacruz (2011) to test students' knowledge about rational numbers. An example screenshot is shown in Figure 2. In this game, students are required to help "Patch," an in-game animal character, move from his starting position to the final destination. In this game, Patch can only move by bouncing on trampolines, and how far Patch can bounce depends on the value of the trampoline. Students can move trampolines onto the grid and energize them by adding coils, which are of whole and fraction unit values. Successful game play requires students

to apply fraction algebra. In this example game, fractional algebra problems are presented in imaginary and engaging set of scenarios for young children. While it is not realistic, it can motivate students to play more and practice their fractional algebra skills. In practice, some game-based assessments also use realistic scenarios, and there is no hard distinction between game-based and simulation-based assessments. Consequently, simulation-based and game-based assessments are often discussed together in the literature (e.g., Kerr, 2015; Kerr & Chung, 2012). From this point onwards, for the sake of simplicity, I will refer to both simulation-based and game-based assessments as *Simulation-Based Assessments*. This terminology is also commonly used in the literature (de Klerk, Veldcamp, & Eggen, 2015).

Figure 2. A screenshot of *Save Patch*.

There are many benefits and values associated with simulation-based assessments. In fact, the pedagogical value of simulation-based assessments has been the focus of several studies. For example, Slotta and Chi (2006) conducted an experimental study to examine the effects of a simulation-based assessment on student learning of electricity. In the experimental group, a simulation-based assessment was used to teach and assess students about electricity, and in the control group, only traditional teaching and assessments were used. They found that students in the experimental group had significantly better learning outcomes (i.e., measured by verbal explanation and problem solutions) than students in the control group. Ioannidou et al. (2010) created a collaborative simulation-based assessment program that required different students to control different simulated human organs. Students needed to understand how different organs worked to achieve homeostasis. The results showed that students in the experimental group significantly outperformed students in the control group in every learning outcome measured and expressed significantly more interest in health. Gee (2008) argued that video or learning games may help students think by promoting embodiment or situated cognition, which is supported by the idea that human thinking is shaped not only by the brain but by the agent's full-bodied and self-regulated interaction with the external world, including many aspects of the body beyond the brain. Shaffer and Gee (2011) further listed many theoretical advantages of simulation-based assessments, including providing information for players to improve; assessing whether a player is ready for future challenges; and integrating learning and assessments.

Besides these pedagogical benefits, simulation-based assessments provide important and unique value in their ability to generate log files, which store the history of everything a student does while using the assessment system (Mayrath, Clarke-Midura, & Robinson, 2011). Log files are also known by many other names including logdata, tracedata, and clickstream data in the

literature. As discussed before, log files contain a rich amount of information about students' problem solving processes, but at the same time, the complexity of log files also poses many challenges for psychometric analysis. In order to utilize simulation-based assessments as formal assessments in practice, educators or test developers need to extract important information from the complex log files and build a psychometric model to infer students' proficiency. This is one reason that *Evidence Centered Design* (ECD; Mislevy, 2006) was proposed and used in modern simulation-based assessments. ECD formalizes the procedures generally undertaken by expert assessment developers (Shute, Hansen, & Almond, 2008), and provides a multi-layered systematic approach to design assessments.

ECD has many components, but from a psychometric perspective the most important component of ECD is the *Conceptual Assessment Framework* (CAF). CAF consists of three models: the student model, the activity model, and the evidence model (de Klerk, Veldcamp, & Eggen, 2015). The *student model* specifies what latent student variables (e.g., proficiency, knowledge, skills, attributes, or misconceptions) educators or test developers intend to measure, and the relationship among these variables. These variables are commonly referred to as the *Student Model Variables* (SMVs). The *activity model* specifies what activities, tasks, or test items are required to elicit student behaviors that reflect SMVs. In the context of simulation-based assessments, the activity model includes all the tasks that are part of a simulation-based assessment. The *evidence model* combines the student model and the task model, and it describes the relationship between SMVs and *Observable Variables* (OVs), which are students' behavioral indicators of the SMVs. It is important to note that the OVs of simulation-based assessments can be quite different from the OVs of traditional assessments. For example, in traditional multiple-choice tests, the OVs are simply the item scores (e.g., 1 for correct and 0 for incorrect). In

simulation-based assessments, the OVs are more complicated. On one hand, log files from simulation-based assessments contain student behavior data that can be scored like traditional assessments. This type of data is often called the *product data* in the literature (de Klerk et al., 2015). An example of product data in a simulation-based assessment would be a student's final solution to a math problem. The scores of product data can be used directly as OVs. On the other hand, simulation-based assessments also store rich information about students' problem solving processes in log files. This type of data is often called *process data* in the literature. Some process data can be used as OVs, while others are irrelevant. The challenge is to identify which part of the process data can be used as OVs for the SMVs. The process of identifying OVs from the raw data is called *evidence identification* in the ECD framework. There are two main ways to identify OVs in simulation-based assessments: expert judgment and data mining (de Klerk et al., 2015). Often both are needed for evidence identification. A more detailed discussion of methods for evidence identification is presented in later sections. After identifying the OVs, the latent SMVs can be estimated using a psychometric model. This process of estimating SMVs from OVs is called *evidence accumulation* in the ECD framework.

The CAF in ECD provides a framework to construct psychometric models from log files. Within this framework, the psychometric properties of simulation-based assessments can be examined. De Klerk et al. (2015) conducted a comprehensive literature review of the psychometric analyses of simulation-based assessments. They identified and reviewed 31 current psychometric studies of simulation-based assessments, and classified these studies into two categories. The first category included studies that used educational data mining techniques to explore and identify students' problem-solving patterns (e.g., Kerr & Chung, 2012). In ECD terms, these studies focus on evidence identification. The second category included studies that

used Bayesian networks (e.g., Mislevy et al., 2001; Shute and Ventura, 2013; Levy, 2014) or traditional psychometric models such as *Item Response Theory* (Quellmalz et al., 2012) and *Confirmatory Factor Analysis* (Quellmalze et al., 2013) to estimate student proficiency. In ECD term, these studies focus on evidence accumulation. It is under the second category, that familiar psychometric properties such as reliability and validity are examined. For example, Quellmalz et al. (2013) showed that simulation-based assessments had better reliability and validity than similar traditional assessments for assessing scientific inquiry. Since the goal of the current study is to explore and identify students' different problem solving patterns in simulation-based assessments, the first category of studies reviewed by De Klerk et al. (2015) is more relevant. Therefore, the following sections focus on reviewing educational data mining techniques to identify students' problem-solving patterns. It is important to note that the literature on educational data mining for simulation based assessment is relatively small, but educational data mining is widely applied in various other e-learning environments – such as Learning Management Systems (e.g., Moodle, eClass, Blackboard, etc), and Intelligent Tutoring Systems (e.g., Cognitive Tutoring System, Koedinger & Corbett, 2006). Although the goals of educational data mining in simulation-based assessments may be slightly different from that of other e-learning contexts, educational data mining techniques from other areas of e-learning can be modified and applied to simulation-based assessments. Thus, in the following subsections, I review studies that examined data mining techniques for the analysis of log files generated in various e-learning environments. Three major categories of data mining techniques are included: cluster analysis, classification (deep learning more specifically), and sequential pattern analysis.

Clustering Techniques in Educational Data Mining

The human brain naturally clusters reality into different groups (Vellido et al., 2011). For example, we naturally cluster animals into different categories (e.g., those that can fly, those that can swim, and those that walk on land) despite the fact that each animal is individually unique. Clustering helps human beings to more efficiently understand and interact with reality by providing a simplified model of reality. However, the human brain is not very efficient at clustering large amount of quantitative data. Therefore, many data mining clustering techniques have been developed to help human beings cluster and understand large sets of quantitative data. In the context of e-learning, the results obtained from clustering students based on their learning or problem solving behaviors can be used to provide specialized feedback to students belonging to each cluster (Zakrzewska, 2008). In addition, researchers can use the clustering results to evaluate and improve the e-learning or assessment software (Kerr, 2015). Furthermore, tools to cluster students based on their learning behaviors may be an efficient way to create online learning groups (Manikandan, Sundaram, & Babu, 2006).

In a very basic and simplified way, data clustering can be defined as assigning each of N data points to one of *K* possible clusters (Vellido et al., 2011). However, modern clustering techniques have more complexity than what this definition implies. There are many different types of modern clustering techniques. Vellido et al. (2011) classified these techniques into hard versus soft, partitional versus hierarchical, and probabilistic versus heuristic. Each of these dichotomies will be discussed shortly. Then, three popular cluster analysis techniques (i.e., *k*-means clustering, finite mixture model, and agglomerative cluster analysis) will be discussed in detail. Lastly, current applications of cluster analysis in e-learning will be reviewed.

Hard versus soft clustering techniques. Hard clustering techniques cluster each response vector (e.g., student) into one of the *K* possible clusters as the basic cluster definition describes. Examples of hard clustering techniques include *k*-means clustering (Vellido et al., 2011) and agglomerative clustering (Everitt, 2011). In contrast, soft clustering techniques assign a probability measure of cluster membership to each data case. For example, suppose we are clustering students based on their problem-solving behaviours. Hard clustering techniques would produce results that specify to which cluster a student belongs (e.g., student A belongs to cluster 2), but soft clustering techniques would produce results that specify the probability a student belongs to each cluster (e.g., student A has a 20% probability of being in cluster 1 but 80% probability of being in cluster 2). Examples of soft clustering techniques include mixture models (McLachlan, 2000) and fuzzy *c*-means cluster (Dunn, 1973).

Partitional versus hierarchical clustering techniques. Partitional clustering techniques assume a single common level for all clusters, while hierarchical clustering techniques assume that the cluster structure is nested or hierarchical. An intuitive example of hierarchical cluster is animal classification, in which animals are clustered into different species. Different species are nested within a genus, which is then nested within a family, and so on. In contrast, partitional clusters do not nest in higher order clusters. Common methods of partitional clustering techniques include *k*-means clustering, mixture model, and self-organizing maps (SOM, Kohonen, 2001), while examples of hierarchical clustering techniques include agglomerative cluster and bisecting *k*-means clustering (Pelleg & Moore, 1999).

Probabilistic versus heuristic clustering techniques. Probabilistic clustering techniques assume data in each cluster reflect a certain probability distribution (e.g., normal distribution or

multinomial distribution). Then, cluster membership probabilities are estimated based on the assumed probability distribution. Mixture models and generative topographic mapping (Bishop, Svensén, & Williams, 1998) are all examples of probabilistic clustering techniques. In contrast, heuristic clustering techniques do not assume that the data reflect any probability distribution, and clustering is based on heuristic algorithms. Examples of heuristic clustering include *k*-means clustering, SOMs, and agglomerative clustering.

Today, there are hundreds of clustering algorithms available. It is beyond the scope of this dissertation to review all of them. In the next subsections, three cluster analysis techniques (i.e., *k*-means clustering, finite mixture model, and agglomerative cluster analysis) will be discussed in detail. These cluster analysis methods were chosen because they represent/demonstrate the different types of cluster analysis previously discussed. *K*-means cluster analysis represents hard, partitional, and heuristic cluster analysis. Finite mixture model represents soft, partitional, and probabilistic cluster analysis. Agglomerative cluster analysis represents hard, hierarchical, and heuristic cluster analysis. Among these methods, *k*-means cluster analysis is the most basic cluster analysis, although finite mixture model is statistically more sophisticated. That being said, the efficiency of *k*-means may be more practical for large datasets.

K-means clustering. The idea of *k*-means clustering was first proposed by Steinhaus (1957). Since then, hundreds of clustering algorithms have been developed, but *k*-means clustering still remains one of the most widely used clustering algorithms due to its simplicity, efficiency, and empirical success (Vellido et al., 2011). As discussed before, *k*-means clustering is a hard, partitional, and heuristic clustering method. The goal of *k*-means clustering is to divide

a data set $D = \{x_1, x_2, \cdots, x_n\}$ into *K* disjoint clusters, $C = \{C_1, C_2, \cdots C_K\}$, where each continuous data case, x_i, is assigned to a unique cluster C_k. Researchers need to define the number of clusters, *K*, arbitrarily. The basic *k*-means clustering algorithm can be summarized as follows:

1. Choose *K* random points as cluster centroids.
2. Assign all observations to their closest cluster centroid according to a distance measure.
3. Re-compute new cluster centroids.
4. Repeat steps 2 and 3 until the cluster membership becomes stable.

According to Vellido et al. (2011), the most commonly used distance measure is the Euclidean distance, which is suitable to detect ball-shaped clusters. Mahalanobis distance can be used to detect ellipse shaped clusters. The strength of *k*-means clustering lies in its simplicity. However, there are two weaknesses with *k*-means clustering. First, researchers need to choose an arbitrary number of clusters, *K*. There is no statistical test to help determine what number of clusters is the most appropriate. Second, *k*-means clustering is sensitive to the starting points of the cluster centroids. In other words, different starting cluster centroids may result in different cluster solutions, as *k*-means clustering does not guarantee that the solution will converge to a global minimum.

Finite mixture models. The initial idea of finite mixture models was proposed as early as 1846 (McLachlan, 2000). However, due to its computational difficulty, the method did not become widely used until the advent of modern computers and the popularization of the *Maximum Likelihood* (ML) parameterization (McLachlan, 1988). Finite mixture models are soft, partitional and probabilistic clustering methods. Finite mixture models assume that the observed data come from a mixture of probability distributions. Each cluster *k* has a probability

distribution $g_k(\vec{x};\vec{\theta}_k)$, where $\vec{x}$ is a p-dimensional random vector, and $\vec{\theta}$ is a vector of parameters of the probability distribution. All finite mixture models can be represented using the general formula:

$$f(\vec{\boldsymbol{x}};\ \vec{\boldsymbol{p}},\vec{\boldsymbol{\theta}}) = \sum_{k=1}^{K} p_k g_k\left(\vec{\boldsymbol{x}};\vec{\boldsymbol{\theta}}_{\boldsymbol{k}}\right), \tag{1}$$

where $f(\vec{\boldsymbol{x}};\ \vec{\boldsymbol{p}},\vec{\boldsymbol{\theta}})$ is the probability distribution of the observed data, and $\vec{\boldsymbol{p}} = (p_1, p_2, \cdots p_K)$ is a vector of cluster proportions. When $\vec{\boldsymbol{x}}$ is continuous, $g_k(\vec{\boldsymbol{x}};\vec{\boldsymbol{\theta}}_{\boldsymbol{k}})$ is often set to be the multivariate normal distribution and $\vec{\boldsymbol{\theta}}_{\boldsymbol{k}}$ includes the mean vector and variance covariance matrix for cluster *k*. This model is often called the *Gaussian Mixture Model* or *Latent Profile Analysis*. When $\vec{\boldsymbol{x}}$ is categorical, $g_k(\vec{\boldsymbol{x}};\vec{\boldsymbol{\theta}}_{\boldsymbol{k}})$ can be set to be the multinomial distribution and $\vec{\boldsymbol{\theta}}$ includes a list of event probabilities. This model is commonly referred to as the *Latent Class Analysis*.

In order to estimate the probability that each data case, $\vec{\boldsymbol{x}}_{\boldsymbol{i}}$, belongs to cluster *k*, the following formula can be used:

$$P(k|\vec{\boldsymbol{x}}_{\boldsymbol{i}}) = \frac{p_k g_k\left(\vec{\boldsymbol{x}}_{\boldsymbol{i}};\vec{\boldsymbol{\theta}}_{\boldsymbol{k}}\right)}{f(\vec{\boldsymbol{x}}_{\boldsymbol{i}};\ \vec{\boldsymbol{p}},\vec{\boldsymbol{\theta}})}. \tag{2}$$

In equation (1) and (2), $\vec{\boldsymbol{p}}$ and $\vec{\boldsymbol{\theta}}$ can be estimated using ML, and the *Expectation Maximization* (EM) algorithm is often used to find solutions to the ML equation. The detailed ML equations and EM algorithm can be found in Everitt, Landau, Leese, and Stahl (2011). However, the finite mixture model is also sensitive to the starting values of $\vec{\boldsymbol{p}}$ and $\vec{\boldsymbol{\theta}}$. In practice, statistical software often randomly initializes many (e.g., 1000) sets of starting values of $\vec{\boldsymbol{p}}$ and $\vec{\boldsymbol{\theta}}$. For each set of starting values, a model is estimated and its log likelihood is computed. The

model with the best log likelihood value will be selected. However, in order to ensure this is indeed the best model, researchers need to check whether the best log likelihood can be replicated with different sets of starting values (Geiser, 2010).

Agglomerative cluster analysis. *K*-means cluster analysis and infinite mixture model both use a top-down approach, which requires the specification of the number of clusters and estimation of cluster centers before assigning cases to clusters. In contrast, agglomerative cluster analysis uses a bottom-up approach, in which cases that are close to each other are combined together to form clusters (Everitt, 2011). More specifically, agglomerative cluster analysis starts with a collection C of N singleton clusters. Each cluster contains one data point. Next, the following procedures are repeated until only one cluster is left: a) compute distances between all clusters and store the distances in a matrix for efficiency; b) find a pair of clusters that has the shortest distance; c) combine the two clusters into a new cluster; and d) remove these two clusters from collection C and add the newly combined cluster into collection C. There are many ways to define the distance between two clusters (c_1, c_2). Using different cluster distances can result in different solutions. Single-link distance is the distance between closest elements in clusters.

$$D_{single\ link}(c_1, c_2) = \min_{x_1 \in c_1,\, x_2 \in c_2} D(x_1, x_2). \tag{3}$$

Using single-link distance tends to produce chain-like clusters. Complete-link distance is the distance between the farthest elements in clusters.

$$D_{complete\ link}(c_1, c_2) = \max_{x_1 \in c_1,\, x_2 \in c_2} D(x_1, x_2). \tag{4}$$

Using complete-link distance tends to produce spherical shaped clusters. Average link distance is the average of all pairwise distances between points in clusters.

$$D_{average\ link}(c_1, c_2) = \frac{1}{|c_1|}\frac{1}{|c_2|}\sum_{x_1 \in c_1}\sum_{x_2 \in c_2} D(x_1, x_2). \quad (5)$$

Using average-link distance has the advantage of being less affected by outliers. Average-link distance is the default distance in SPSS. Centroid distance is the distance between centroids of two clusters.

$$D_{centroid\ link}(c_1, c_2) = D\left(\left(centroid_{c_1}\right), \left(centroid_{c_2}\right)\right). \quad (6)$$

Centroid distance is the most intuitive approach. Finally, Ward's distance is the sum of squared distance between all points shared in the two clusters and the centroid of the two clusters.

$$TD_{c_1 \cup c_2} = \sum_{x \in c_1 \cup c_2} D\left(x, \mu_{c_1 \cup c_2}\right)^2. \quad (7)$$

Ward's distance uses the same objective function that *k*-means used. To efficiently update distance matrix in agglomerative cluster analysis with any type of cluster distance, Lance and Williams' (1967) formula is used, which is shown below.

$$d_{k(ij)} = \alpha_i d_{ki} + \alpha_j d_{kj} + \beta d_{ij} + \gamma\left|d_{ki} - d_{kj}\right|, \quad (8)$$

where $d_{k(ij)}$ is the distance between cluster k and a newly formed cluster by combining previous cluster i and j; α_i, α_j, β, and γ are coefficients that determine the type of cluster distance; d_{ki}, d_{kj}, and d_{ij} are distances between cluster k and i, cluster k and j, and cluster i and j, respectively. The coefficients associated with different types of cluster distances are presented in Table 1.

Table 1. *Lance* and *Williams* (1967) *Formula Coefficients for Different Cluster Distances*.

Method	α_i	α_j	β	γ
Single link	.5	.5	0	-.5
Complete link	.5	.4	0	.5
Average link	$\frac{n_i}{n_i + n_j}$	$\frac{n_j}{n_i + n_j}$	0	0

Centroid Link	$\frac{n_i}{n_i + n_j}$	$\frac{n_j}{n_i + n_j}$	$\frac{-n_i n_j}{(n_i + n_j)^2}$	0
Ward distance	$\frac{n_i + n_k}{n_i + n_j + n_k}$	$\frac{n_j + n_k}{n_i + n_j + n_k}$	$\frac{-n_k}{n_i + n_j + n_k}$	0

$n_i, n_j, and\ n_k$ are the sample sizes for cluster i, j, and k.

Agglomerative cluster analysis produces a dendrogram that shows which clusters are combined in each iteration. An example of dendrogram of a five-case agglomerative cluster is shown in Figure 3. Researchers can choose the number of clusters based on criteria discussed in the next section.

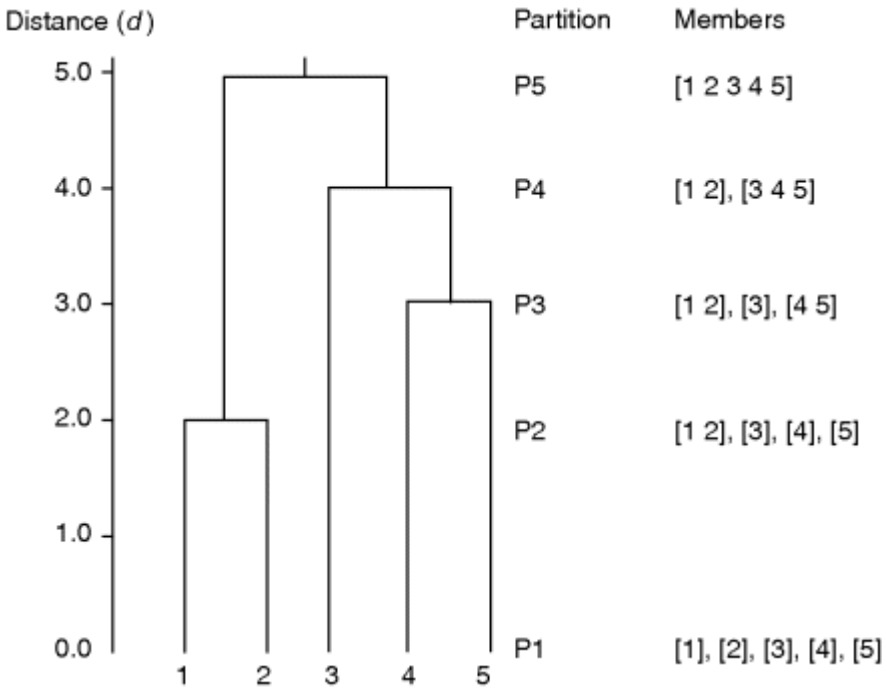

Figure 3. Example Agglomerative Cluster Dendrogram of five cases. Reprinted from *Cluster Analysis* (5th Edition), by Everitt, B.S., (2011), Chichester, West Sussex, U.K: Wiley.

Determining the number of clusters. Many cluster analysis techniques (e.g., *k*-means clustering and finite mixture model) require the specification of the number of clusters, *k*, before model estimation. While other methods – such as different variations of hierarchical cluster analysis – do not require the number of clusters to be known before model estimation, the

number of clusters still needs to be determined after model estimation to interpret the results in a meaningful way. The correct choice of *k* is often ambiguous. Various methods have been suggested. Each method is based on unique assumptions and criteria, and may be helpful in particular situations. As recommended by Everitt (2011), researchers should not depend on a single rule for selecting the number of clusters, and instead synthesize the results of several techniques. In the following subsections, four popular approaches to determine the number of clusters are discussed.

Determining clusters graphically. Humans are good at discerning visual patterns. As Everitt (2011) suggested, "graphical views of multivariate data are important in all aspects of their analysis." When the data are continuous and have one or two dimensions, it is relatively easy to visualize the data using scatter plot. However, when the data have more than two dimensions, visualization is less straightforward. There are several approaches to reduce the dimensions of the data for visualization (Everitt, 2011). *Principal Component Analysis* (PCA) is the most basic and common approach. In PCA, eigenvalues and eigenvectors of the variance covariance matrix or correlation matrix are computed. The eigenvalues represent the variances extracted by each component. The eigenvectors are coefficients that can be used to combine the variables linearly to obtain the principal components. The first component extracts the largest amount of variance from the data. The second component is orthogonal to the first component, and it extracts the second largest amount of variance from the data. Using the first two components, multivariate data can be visualized, and the number of clusters can be determined visually. The disadvantage of this approach is that the first two components may not capture enough variance to account for the variability of the data.

Information criterion approach. The information criterion approach uses relative model fit indices to determine the number of clusters. There are two commonly used information criteria: *Akaike Information Criterion* (AIC; Akaike, 1998), *Bayesian Information criterion* (BIC; Schwarz, 1978), and sample size adjusted BIC. All these information criteria consider the probability of observing the data given the assumption that a cluster model is true. In other words, these criteria involve using the model-implied probability distribution of the dependent variables. This means that researchers need to specify the probability distribution of the data within each cluster. Often, Gaussian distribution is used to model continuous data (e.g., Goutte, Hansen, Liptrot, & Rostrup, 2001). Consequently, the accuracy of these criteria depends on whether the data indeed follow the specified probability distributions. With this understanding, AIC and BIC are introduced in the following paragraphs.

In AIC, the difference (measured using Kullback-Leibler divergence) between a model implied probability distribution and the true population distribution is estimated. More specifically, the AIC formula is:

$$AIC = 2k - 2\log(\hat{L}), \tag{9}$$

where k is the number of estimated parameters in the model and $\hat{L}$ is the maximum value of the likelihood function for the model. Models with lower values of AICs are considered better because it means they are similar to the population model. In order to use AIC to determine the number of clusters, models with different number of clusters are run. For each model, AIC is calculated. The number of cluster that gives the lowest value of AIC is chosen.

In BIC, the probability that a cluster model is true given the data is estimated using the Bayes' Rule, which involves the use of the model-implied probability distribution of the data. In

order to be consistent with AIC, a negative sign is applied to this estimated probability, so that lower BIC value represents better model fit. More specifically, the BIC formula is:

$$BIC = \log(n)\, k - 2\log(\hat{L}), \tag{10}$$

where *n* is the number of data points, *k* is the number of parameters in the model, and $\hat{L}$ is the maximum value of the likelihood function for the model. To determine the number of clusters, models with different number of clusters are run, and the one with the lowest BIC is chosen.

Within cluster distance approach. As the name suggests, this approach depends on calculating the sum of the within-cluster distances between every data point in a cluster and the cluster centroid. The total within-cluster distance is:

$$W_K = \sum_{k=1}^{K} \sum_{i=1}^{n_k} \|x_{ki} - \mu_k\|^2, \tag{11}$$

where *k* is the number of clusters, n_k is the number of cases in cluster *k*, x_{ki} is a point in cluster *k*, and μ_k is the cluster *k*'s centroid. Intuitively, if the true number of clusters is *k*, then setting the number of clusters to be smaller than *k* will lead to much larger W_K, and setting the number of clusters to be greater than k will only lead to minor decrease in W_K. Based on this intuition, several statistics have been developed. Tibshirani, Walther, and Hastie (2001) proposed the *gap statistic*, which attempts to standardize the comparison of $\log(W_K)$ with a null reference distribution with no obvious clustering. Most of the time, the null distribution is a uniform distribution with the maximum and minimum values from the data as the boundaries. More specifically, the gap statistic is defined as:

$$Gap(k) = E[\log(W_K^*)] - log(W_K). \tag{12}$$

To estimate $E[\log(W_K^*)]$, B copies of reference datasets are generated by sampling uniformly from the original dataset's bounding box (i.e., maximum and minimum values). For each dataset, $\log(W_K^*)$ is calculated, then the average of these $\log(W_K^*)$ is used to estimate $E[\log(W_K^*)]$. These $\log(W_K^*)$ have a standard deviation *sd(K)*, which is biased by simulation error. After correcting the errors, it equals to

$$s_K = \sqrt{1 + \frac{1}{B}} sd(K). \tag{13}$$

Finally, the optimal number of clusters is the smallest *k* such that $Gap(k) \geq Gap(k+1) - s_{k+1}$.

Pham, Dimov, and Nguyen (2005) proposed the *f(K)* statistic, which is defined as follows:

$$f(K) = \begin{cases} 1 & if\ K = 1 \\ \dfrac{W_k}{\alpha_k W_{k-1}} & if\ W_{k-1} \neq 0, \forall K > 1 \\ 1 & if\ W_{k-1} = 0, \forall K > 1 \end{cases}$$
$$\alpha_K = \begin{cases} 1 - \dfrac{3}{4p} & if\ K = 2\ and\ N_d > 1 \\ \alpha_{K-1} + \dfrac{1 - \alpha_{K-1}}{6} & if\ K > 2\ and\ N_d > 1 \end{cases} \tag{14}$$

where *p* is the number of variables in the data set and α_K is a weight factor. In order to determine the number of clusters, models with different number of clusters are run and the one with the lowest *f(K)* is chosen.

Significance tests. For finite mixture models, there are several statistical tests available to help determine the number of clusters. Commonly-used tests include the bootstrap likelihood ratio difference test (Langeheine, Pannekoek, & van de Pol, 1996; von Davier, 1997) and the Vuong-Lo-Mendell-Rubin test (Lo, Mendell, & Rubin, 2001). According to a simulation study

conducted by Nylund, Asparouhov, and Muthén (2007), the bootstrap likelihood ratio test is considered as more accurate than the Vuong-Lo-Mendell-Rubin test. Thus, only the bootstrap likelihood ratio test will be reviewed here. In the bootstrap likelihood ratio test, the original data are sampled with replacement many times (e.g., 1000). For each dataset, a likelihood ratio is calculated between *k* and *k-1* class. Likelihood ratio is defined by the following formula:

$$likelihood\ ratio = -2\left[\log \hat{L}(\theta_k) - \log \hat{L}(\theta_{k-1})\right], \quad (15)$$

where $\hat{L}(\theta_k)$ is the maximum likelihood value of model with *k* classes and $\hat{L}(\theta_{k-1})$ is the maximum likelihood value of model with *k-1* class. Finally, the *p* values of the likelihood ratio difference are estimated based on all the bootstrap sampled datasets. To determine the number of clusters, keep running models with more clusters until the bootstrap likelihood ratio is no longer significantly different. The cluster number is set to be *k-1*.

Applications of cluster analysis in educational data mining. Since the development of e-learning in the late 1990s, cluster analysis has been used in many educational data mining studies. In order to summarize these studies efficiently, three major aspects of these studies will be discussed: purposes of the studies, feature selection methods, and clustering analysis methods.

Purposes. In the context of e-learning, the most common purpose for using clustering analysis is to personalize students' learning experience (Zakrzewska, 2008; Mylonas, Tzouveli, & Kollias, 2007; Lu et al., 2007; Tian, Wang, Zheng, & Zheng, 2008). Personalization stems from the idea that a good e-learning program should be able to adapt to different students' learning preferences. However, since each student is unique, it is not feasible to create an e-learning program that can adapt to each individual student. In order to deal with this problem,

students are clustered into groups based on their learning preferences and even personality. Then, the e-learning program can provide a specific learning environment for each group of students with common characteristics. It is believed that personalization can help improve students' motivation and learning outcomes (Zakrzewska, 2008).

Another common application of cluster analysis in e-learning is to create online study groups (Tang & Chan, 2002; Christodoulopoulos & Papanikolaou, 2007; Talavera & Gaudioso, 2004). Group learning is an effective way to promote greater academic achievement (Tang & Chan, 2002). However, the efficiency of group learning depends on the characteristics of the group members. While there are different theories on the effectiveness of group composition, cluster analysis can help create both homogeneous and heterogeneous groups based on students' learning styles, and knowledge levels (Christodoulopoulos & Papanikolaou, 2007).

In addition to personalization and group formation, cluster analysis has also many diverse applications in e-learning including enhancing prediction (Khalil, Li, & Wang, 2008; Romero, González, Ventura, del Jesus, & Herrera, 2009); assessing motivation or ability (Hershkovitz & Nachmias, 2011; Mullier, 2003); classifying novice programmer errors (Sison, Numao, & Shimura, 2000); detecting circumstances under which students accessed help (Nilakant & Mitovic, 2005); and detecting service usage patterns (Harpstead et al. 2013).

In the context of simulation-based assessments, cluster analysis has been used to analyze student solution strategies and error patterns (Kerr & Chung, 2012). Cluster analysis has also been used to improve simulation-based assessment design and visualize changes in problem-solving strategies used across attempts (Kerr, 2015). Although the purposes of these studies seem

different, they all used cluster analysis to identify students' problem-solving patterns as a first step.

Feature selection methods. In order to apply cluster analysis to analyze log files, variables of interest must be first extracted from the log files. This process is called feature selection. A majority of the clustering studies have used action frequency and sometimes reaction time in log files as the input variables (e.g., Teng, Lin, Cheng, & Heh, 2004; Kerr, 2015; Rodrigo, Anglo, Sugay, & Baker, 2008; Amershi & Conati, 2011). In these studies, researchers need to determine which actions are important based on theory and/or qualitative analysis of a small sample of the log files. As in e-learning environments, different actions may reflect distinct levels of importance in the clustering process, action frequency variables are often normalized to have a value between 0 and 1 (Zhang, Cui, Wang, & Sui, 2007; Teng, Lin, Cheng, & Heh, 2004). Besides frequency and reaction time, psychological questionnaires are often used to supplement the log file clustering (Zakrzewska, 2008; Buckley, Gobert, & Horwitz, 1999). Most of these questionnaires measure students' learning preferences, motivation, and personality.

Cluster analysis methods. In the sample of studies reviewed, *k*-means clustering still remains one of the most commonly used clustering methods (Khalil, Li, & Wang, 2008; Manikandan, Sundaram, & Babu, 2006; Rodrigo, Anglo, Sugay, & Baker, 2008; Harpstead et al., 2013; Amershi & Conati, 2011). As explained before, this is mostly due to *k*-means clustering's simplicity and computational efficiency. Especially, when the data set is large, computational efficiency may outweigh the advantages of other clustering methods. Fuzzy clustering methods such as fuzzy *c*-means clustering are also very commonly used in e-learning clustering studies (Christodoulopoulos & Papanikolaou, 2007; Kerr, 2015; Lu, Li, & Liu, 2007; Tian, Wang,

Zheng, & Zheng, 2008). Fuzzy clustering methods are often preferred when researchers expect there to be significant overlap between the clusters (Kerr, 2015). In this case, estimating each data case's probability of belonging to each cluster is more informative than assigning each data case exclusively to one cluster. Besides these two popular clustering methods, various other clustering methods have been used in the e-learning literature, including hierarchical clustering (Mylonas, Tzouveli, & Kollias, 2007); self-organizing maps (Mullier, 2001); generative topographic mapping (Vellido, Castro, Nebot, & Mugica, 2006); and other newly proposed and less well-known clustering algorithms, such as matrix based clustering and incremental relational clustering (Zhang, Cui, Wang, & Sui, 2007; Sison, Numao, & Shimura, 2000). The final number of clusters is often determined based on the meaningfulness of interpretation (Kerr, 2015; Etheredge, Lopes, & Bidarra, 2013).

Overall, the literature review of clustering studies in e-learning suggests that there are many studies that examine how to cluster students based on their learning preferences and activity frequencies in log files; and *k*-means and fuzzy clustering are currently the most popular clustering methods. However, all current clustering studies ignore the sequence or order of students' actions. Consequently, the current clustering methods may not be optimal for studying students' problem solving strategies because the sequence of what a student does during problem solving may play an important role in determining the effectiveness, strengths, and weaknesses of different problem-solving strategies.

Classification in Educational Data Mining

Classification means assigning an object into a category based on the object's characteristics (Hämäläinen, & Vinni, 2011). The term "classification" can lead to

misunderstanding in two ways. First, classification may sound similar to clustering, but the difference is that classification is learned through *labelled* data, while clustering is learned through *unlabeled* data. For example, when we learn to classify animals, we need to label which animals are dogs and which animals are cats in the data. In contrast, when we cluster animals, we can group animals based on any perceived similarity without the need of knowing any animal names. Second, the term classification often implies that the classification outcome variable is a categorical variable. However, in the context of educational data mining, the outcome variable can also be a continuous variable. Thus, classification can be seen as association and/or prediction.

There are many applications of classification in educational data mining. Hämäläinen, and Vinni (2011) reviewed the literature and found three common areas of application. First, classification algorithms have been used to predict students' academic performance or course outcomes. For example, Minaei-Bidgoli, Kashy, Kortemeyer, and Punch (2003) applied a genetic algorithm to log data generated from a web-based learning program to predict students' academic performance. The results suggested that the genetic algorithm displayed over 90% accuracy at predicting student success/failure. This type of study is useful for identifying students who are at risk early in the course. Second, classification algorithms can be used to predict a student's success in subsequent tasks. For example, Desmarais and Pu (2005) compared Bayesian networks, one of the classification algorithms, and item response theory in terms of accurately predicting students' success in follow-up items based on their previous responses, and they found that Bayesian network revealed superior performance. Third, classification algorithms can be used to predict students' strategy and metacognition. For example, Barker, Trafalis, and Rhoads (2004) applied a latent response model to log files generated from an Intelligent Tutor

program to predict whether students were gaming the system by abusing the flaws of the intelligent tutor program to pass different learning tasks. Fourth, classification algorithms can be used to predict students' motivation. For example, Cocea and Weibelzahl (2006) applied a decision tree algorithm to log files generated from a web-based learning program to predict students' motivation, and they found that the time students spent reading online was the best predictor of motivation.

As can be seen, various classification algorithms are used in educational data mining. However, the most common classification algorithms include regressions, decision trees, Bayesian classifiers, support vector machines, k-nearest neighbor classifiers, and artificial neural networks (for a more detailed description of these algorithms see Witten and Frank, 2005). Each algorithm also has many variations. Most, if not all, of these classification algorithms are adapted from the field of data mining and machine learning.

At this point, I want to take a slight detour from educational data mining, and discuss a general research trend in the context of data mining and machine learning. That is, despite the diversity of classification algorithms, recently, one classification approach, *deep learning neural networks*, has become dominant in the field of machine learning and data mining. Deep learning neural networks consistently outperform other classification algorithms in different areas of applications, such as computer vision and automatic speech recognition (Ciresan, Meier, & Schmidhuber, 2012; Hinton et al. 2012; Deng, Hinton, & Kingsbury, 2013). In the field of image recognition, for example, Ciresan et al. (2012) compared the prediction error rates of their deep learning neural networks with other previous state-of-the-art algorithms at classifying various types of images, including handwritten digits, letters, Chinese characters, 3D model images,

traffic signs, and colored images of objects. They found that their deep learning neural network had significantly lower prediction error rates than other classification algorithms, and even outperformed human beings in traffic sign recognition. Similarly, in the field of speech recognition, Hinton et al. (2012) summarized the findings of four top research groups (i.e., University of Toronto, Microsoft research, Google, and IBM Research). The results showed that deep learning neural networks had significantly lower prediction error rates than previous state-of-the-art-algorithms in various speech recognition tasks. Thus, rather than reviewing every classification approach, I will focus on reviewing deep learning neural networks in the following sections. First, a brief introduction to deep learning neural networks is provided. Then two popular deep learning models, the *feedforward neural network* and *the recurrent neural network*, are discussed in detail. Lastly, applications of deep learning in the field of educational data mining are reviewed.

Deep learning. According to Deng and Yu (2014, p. 199), deep learning neural networks can be formally defined as "a class of machine learning techniques that exploit many layers of non-linear information processing for supervised or unsupervised feature extraction and transformation, and for pattern analysis and classification". Intuitively and simply, a deep learning neural network can be seen as an artificial neural network with multiple hidden layers. The earliest deep learning algorithm was proposed by Ivakhnenko and Lapa (1965). Between the 1970s and 1990s, various deep learning neural networks were developed (e.g., Ivakhnenko, 1971; Fukushima, 1980; Hinton, Dayan, Frey, & Neal, 1995). However, due to the slow computational speed of early computers and the lack of large amounts of data to train the networks, these deep learning neural networks did not become popular until later in the 2000s. In 2006, Hinton and Salakhutdinov (2006) published an influential paper that explained how to train deep learning

neural networks with an algorithm called greedy layer-wise pretraining. Although similar algorithms had been proposed earlier by Schmidhuber (1992), the timing of Hinton and Salakhutdinov's (2006) publication led to a resurgence of interest in deep learning neural networks. Since this resurgence, deep learning neural networks have become part of many state-of-the-art systems in various disciplines (Ciresan et al., 2012; Hinton et al. 2012; Deng et al., 2013). Besides having higher prediction accuracy than other classification algorithms, another important advantage of deep learning neural networks is that they can automatically extract meaningful features from raw data, and consequently depend very little on human expertise to extract features as inputs (LeCun, Bengio, & Hinton, 2015). Recently, deep learning has been applied in the field of educational data mining (e.g., Piech et al., 2015) to model students' learning over time. In order to gain a more detailed understanding of deep learning, two common types of deep learning neural networks (i.e., Deep Feedforward Neural Network and Long Short Term Memory Model) are discussed in the following sub-sections.

Deep Feedforward Neural Networks. As mentioned earlier, a deep learning network can be viewed as an artificial neural network with multiple hidden layers. Thus, in order to understand deep feedforward neural networks, it is first necessary to understand feedforward artificial neural networks. The feedforward artificial neural network is the most basic or archetypal artificial neural network. It is also referred to as the *Multilayer Perceptron* (Goodfellow, Bengio, & Courville, 2016). Intuitively, it is loosely inspired by the human brain. Statistically, it can be seen as an extension of logistic regression. The conceptual diagram of a feedforward neural network is shown in Figure 4.

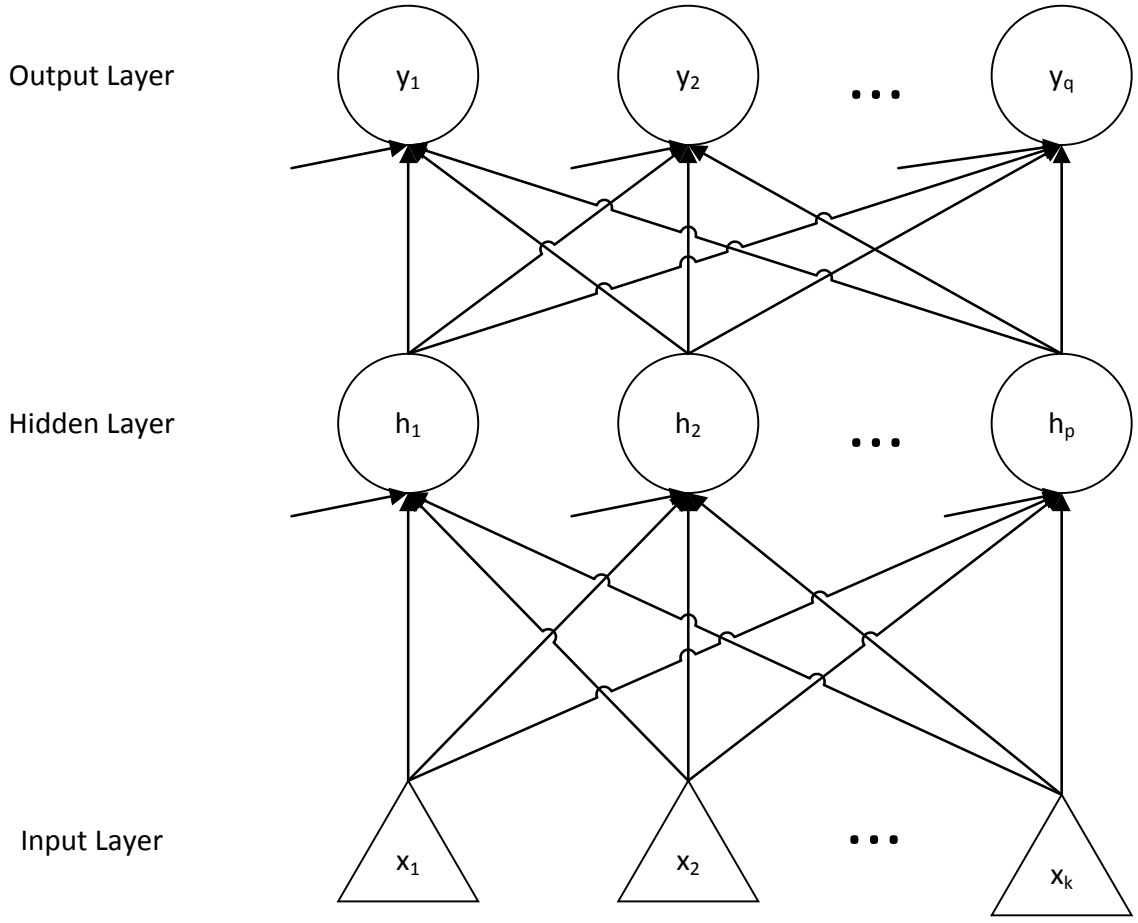

Figure 4. The conceptual diagram of a feedforward neural network.

In Figure 4, the model network has three layers of units: the input layer (x), the hidden layer (h), and the output layer (y). The input layer represents the raw input data; the hidden layer represents important features the network extracts from the raw data, which are then used to predict the outcome (LeCun et al., 2015). Each triangle in the input layer represents an input unit or independent variable. Each circle in the hidden and output layers represents a processing unit, called a *neuron.* (In the following paragraphs, I will use the terms "neuron" and "unit" interchangeably.) A hidden neuron first linearly combines inputs from the previous layer using *weights* stored in the connecting arrows, and then transforms the linearly combined inputs using a non-linear function, called an *activation function*. The most commonly used activation function

for hidden neurons is the *sigmoid function*, which is also called the *logistic function*. Thus, each neuron can be seen as a logistic regression unit. For output neurons, researchers can choose the activation function based on the characteristics of the output variables. If the output variables are continuous, then no activation function is necessary for the output units. If the output variables are dichotomous, then sigmoid function can be chosen. If the output variables are categorical, then *softmax function* can be chosen as the activation function for the output units (Goodfellow et al., 2016). More formally, a feedforward neural network with sigmoid activation functions can be represented using the following equation:

$$y = f(x) = sigmoid\left(\vec{\boldsymbol{b}}^{(2)} + \boldsymbol{W}^{(2)}\left(sigmoid\left(\vec{\boldsymbol{b}}^{(1)} + \boldsymbol{W}^{(1)}\vec{\boldsymbol{x}}\right)\right)\right), \quad (16)$$

where $sigmoid(t) = \frac{1}{1+exp(-t)}$, $\vec{\boldsymbol{b}}^{(1)}$ and $\vec{\boldsymbol{b}}^{(2)}$ are the intercept (also called bias) vectors of the hidden layer and output layer respectively, $\boldsymbol{W}^{(1)}$ and $\boldsymbol{W}^{(2)}$ are the weight matrices for the hidden layer and output layer respectively, x is a vector of input variables, and y is a vector of output variables. Note that if there is no hidden layer in the model, or in other words, the input layer is directly connected to the output layer, then equation (16) reduces to $f(x) = sigmoid\left(\vec{\boldsymbol{b}}^{(1)} + \boldsymbol{W}^{(1)}\vec{\boldsymbol{x}}\right)$, which is mathematically equivalent to a multivariate logistic regression. The bias and weight parameters (i.e., $\vec{\boldsymbol{b}}^{(1)}$, $\vec{\boldsymbol{b}}^{(2)}$, $\boldsymbol{W}^{(1)}$ and $\boldsymbol{W}^{(2)}$) in equation 16 can be estimated using ML, and optimized using some variants of the gradient descent algorithm (e.g., rmsprop, and adadelta; see Goodfellow et al., 2016).

Based on the above description of a feedforward neural network, a *deep* feedforward neural network can be simply understood as a feedforward neural network with more than one layer of hidden units. In general, the more hidden layers and hidden units a network has, the

more powerful the network is. For simple classification problems, however, one hidden layer is enough. Adding more hidden layers will create unnecessary computational burdens and increase computational time for a simple problem. For more complex problems, more hidden layers and hidden units can be added to capture the complexity of the data. There is no hard rule to determine the number of hidden layers and hidden units. In general, researchers can keep experimenting with the number of hidden layers and units until the network produces satisfactory accuracy for a specific classification problem (Goodfellow et al., 2016). However, one common optimization problem in deep learning is overfitting, which occurs when the deep learning network has high prediction accuracy for the training data, but low prediction accuracy for testing data. Here, training data are the data used to estimate the parameters of the network, and testing data are new data used to test the trained network's prediction accuracy. In order to deal with the problem of overfitting, *weight decay regularization* (Goodfellow et al., 2016, p. 231-234), the *greedy layerwise pretraining algorithm* (Hinton & Salakhutdinov, 2006), and the *dropout* algorithm (Hinton, Srivastava, Krizhevsky, Sutskever, & Salakhutdinov, 2012) can be used. The basic idea of weight decay regularization is to make the overall length of the weight matrix (stringed into a vector) as small as possible when performing ML. Consequently, the weights of those hidden units that are less useful in predicting the outputs are forced to decay to 0. In other words, weight decay regularization allows hidden units to compete with each other, and allows only the more impactful hidden units to have non-zero weights. Since extra hidden unit weights are forced to reduce to zero, extra hidden units cease to cause the problem of overfitting. There are many variations of weight decay regularization (for a detailed discussion see Goodfellow et al., [2016, p. 231-234]). The greedy layerwise pretraining algorithm is a method to initialize the network parameter values. Having meaningful starting parameter values

will influence the meaningfulness of the features extracted by the hidden units. It first uses unsupervised learning (e.g., algorithms similar to principal component analysis) to initialize the weights in each layer, then it uses maximum likelihood and gradient descent to train the network. Dropout is a simpler algorithm designed to prevent overfitting. It works by randomly deleting half of the hidden units during each parameter estimation cycle. This is to ensure the hidden units represent independently meaningful features. The detailed optimization algorithms are not the focus of this dissertation, and they can be found in the original papers and Goodfellow et al.'s (2016) textbook on deep learning.

Long Short Term Memory Model. Feedforward neural networks are good for classifying cross sectional data (i.e., data collected at one time point), but they are not designed for classifying sequential or longitudinal data (i.e., time series). In order to analyze sequential or longitudinal data, recurrent neural networks were developed (Goodfellow et al., 2016). The major difference between a feedforward neural network and a recurrent neural network is that in a recurrent neural network, a hidden unit not only influences the output units, but it also influences itself. More specifically, this means a hidden unit's state at time *t* depends on its (and other hidden units') previous states. A conceptual diagram of a recurrent neural network is shown in Figure 5(a). For simplicity, each of the input, hidden, and output units in Figure 5(a) contains a vector of values rather than a single value as shown in Figure 4, and the connection arrows contain a matrix of weights rather than a single value of weight. There is another way to represent the recurrent neural network: the network can be unfolded over time as in Figure 5(b). In Figure 5(b), *t* represents time. As can be seen, input at each time point is fed into the network sequentially. Hidden unit values at time *t*, vector $h_{t,}$ depend on their previous values, vector h_{t-1},

and similarly vector h_{t+1} depends on vector h_t. The output values at time *t*, vector $y_{t,}$ directly depend on hidden unit values at time t, vector h_t, but also indirectly depend on vector h_{t-1} and x_{t-1}.

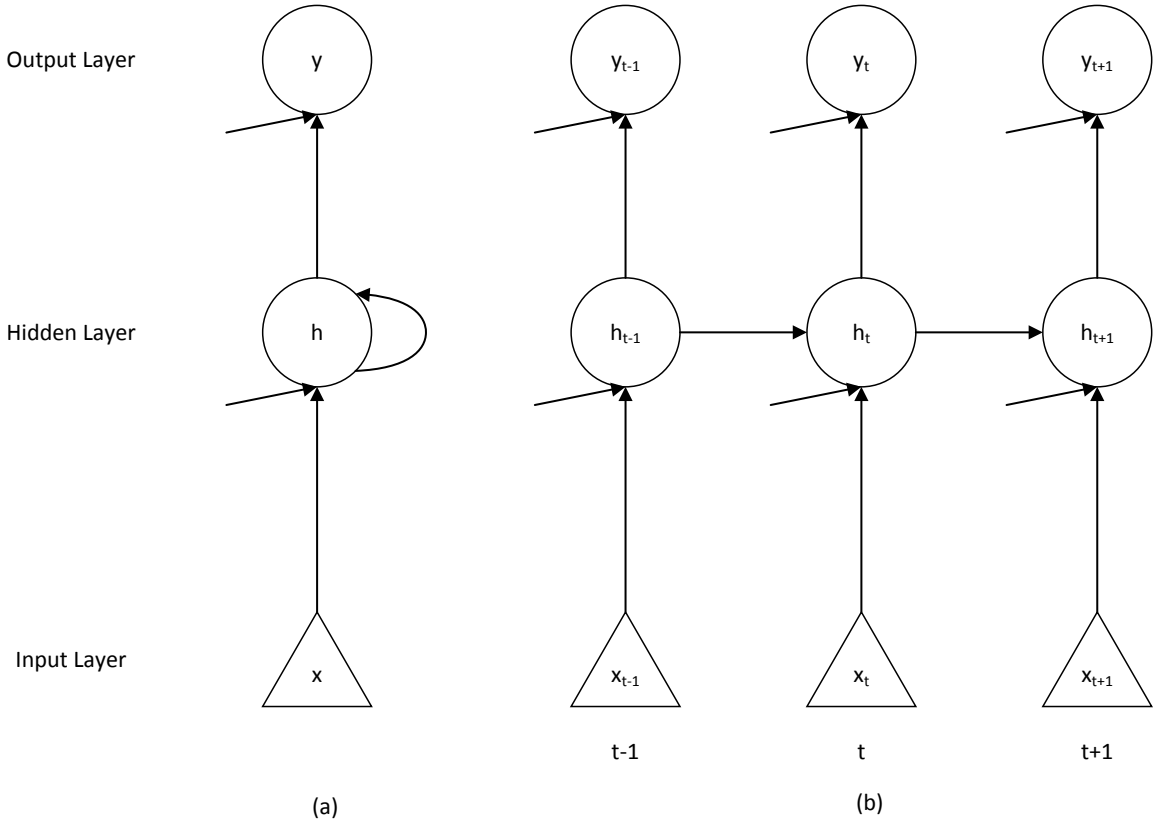

Figure 5. Two ways to represent a recurrent neural network: (a) a folded representation of a recurrent neural network, and (b) an unfolded over time representation of (a).

There are many variations of recurrent neural networks. Currently, one of the most popular and powerful recurrent neural networks is the *Long Short Term Memory Model* (LSTM; LeCun, Bengio, & Hinton, 2015). LSTM was first proposed by Hochreiter and Schmidhuber (1997). Similar to a traditional recurrent neural network, LSTM hidden units also depend on their previous states, and the network can be unfolded overtime. However, LSTM has specialized hidden layers that allow the network to learn to ignore irrelevant input information, retain useful information over time, and forget information once it is no longer useful. Most importantly,

LSTM solves the optimization problem of diminishing gradient, which causes the traditional recurrent network to be unable to retain information for a long time (for more detailed mathematical definition and proof of the diminishing gradient problem and how LSTM resolves it, the reader should consult Hochreiter and Schmidhuber, 1997). The next sub-section focuses on LSTM's hidden layer, commonly referred to as the *LSTM unit*.

Unlike a feedforward or traditional recurrent neural network hidden unit, information does not directly flow into and out of a LSTM unit. A LSTM unit has "gates" that determine whether information should be inputted, outputted or retained for the next time step (t+1). A conceptual diagram of a LSTM network is shown in Figure 6.

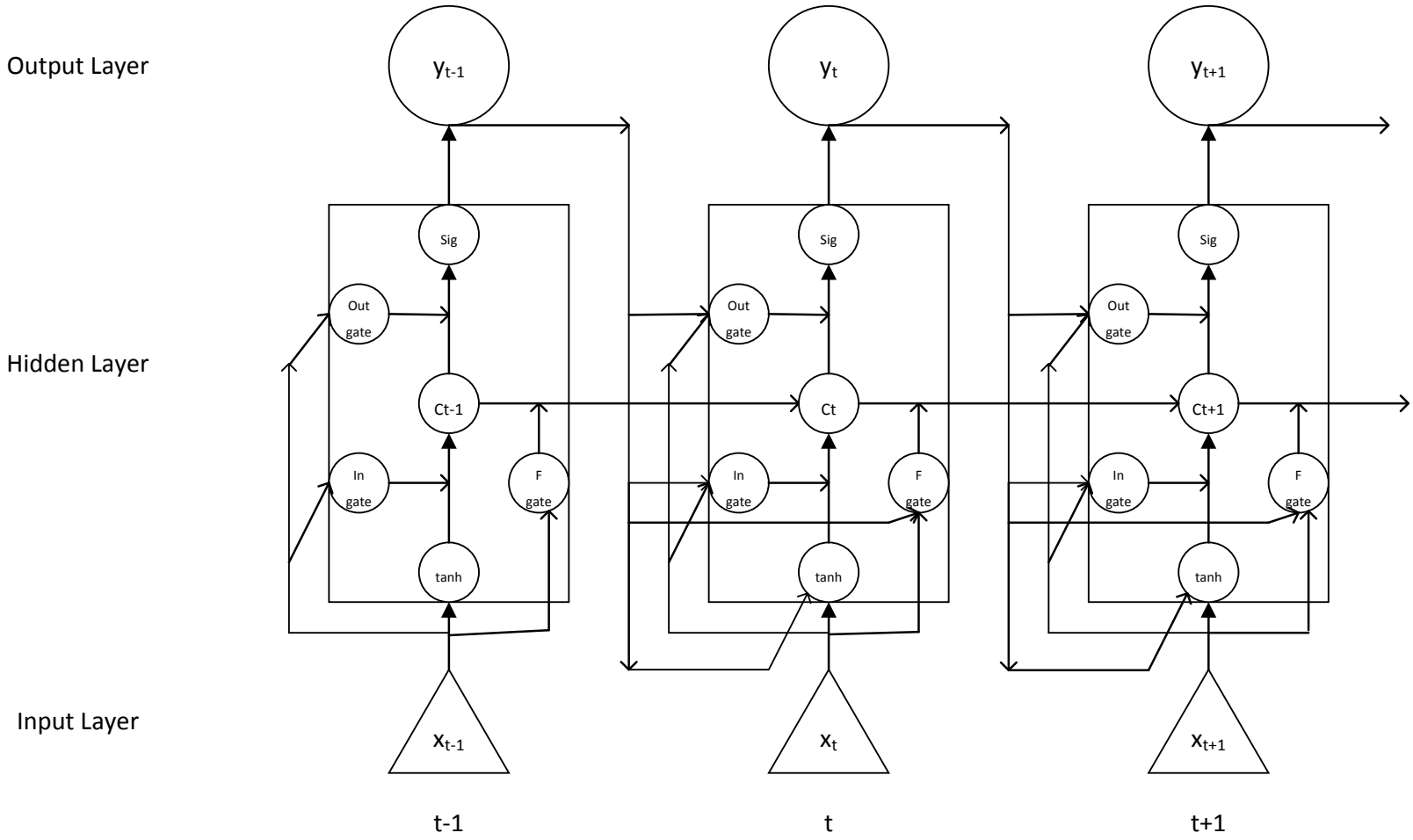

Figure 6. A conceptual diagram for a LSTM network unfolded over time.

As can be seen, an LSTM network has similar input and output layers as a feedforward neural network. However, an LSTM network has a more complex hidden layer, which is called a LSTM unit. Inside an LSTM unit, there are several smaller processing units called *gates*. These gates control when and how much information flows in and out of the LSTM unit. In the center of a LSTM unit, there is a cell that stores hidden unit values in a vector. The relationships among the gates, cell, inputs, and previous LSTM unit outputs can be understood by examining the equation for each gate and cell. The equation for the input gate is:

$$\vec{\boldsymbol{i}}_{\boldsymbol{t}} = sigmoid\left(\boldsymbol{W}_{\boldsymbol{i}}\vec{\boldsymbol{x}}_{\boldsymbol{t}} + \boldsymbol{U}_{i}\vec{\boldsymbol{h}}_{\boldsymbol{t-1}} + \vec{\boldsymbol{b}}_{\boldsymbol{i}}\right), \tag{17}$$

where $\vec{\boldsymbol{i}}_{\boldsymbol{t}}$ is the input gate's value at time t, $\boldsymbol{W}_{i}$ is the weight matrix for input units at time t, $\vec{\boldsymbol{x}}_{\boldsymbol{t}}$ is the input vector at time t, $\boldsymbol{U}_{i}$ is the weight matrix for LSTM unit outputs at time t-1, and $\vec{\boldsymbol{b}}_{\boldsymbol{i}}$ is a vector of intercept or bias. The input gate first linearly combines input at time t and LSTM unit outputs at time t-1, and then transforms the linear combination using the sigmoid function. Consequently, the input gate's output value ranges between 0 and 1. When it is equal or close to 0, the gate can be considered closed, and no information will be able to flow into the cell of the LSTM unit. When it is equal or close to 1, the gate can be considered open, and the input information will be able to flow into the cell of the LSTM unit. The total input information of a LSTM unit is defined as follows:

$$\hat{\vec{C}}_{t} = tanh\left(\boldsymbol{W}_{c}\vec{x}_{t} + \boldsymbol{U}_{c}\vec{h}_{t-1} + \vec{b}_{c}\right), \tag{18}$$

where $\hat{\vec{C}}_{t}$ is the total input into the LSTM unit, *tanh* is the hyperbolic tangent function, $\boldsymbol{W}_{c}$ and $\boldsymbol{U}_{c}$ are the weight matrices for $\vec{x}_{t}$ and $\vec{h}_{t-1}$ respectively, and $\vec{b}_{c}$ is the intercept vector. Equation 18 shows that the total input of an LSTM unit not only includes the input values at time t, but also includes the LSTM unit output values at the previous time step. Beside the input gate, there

is also a forget gate, which determines whether previous stored information should be erased. The forget gate equation is shown as follows:

$$\vec{f}_t = sigmoid\left(\boldsymbol{W}_f\vec{\boldsymbol{x}}_{\boldsymbol{t}} + \boldsymbol{W}_f\vec{\boldsymbol{h}}_{\boldsymbol{t-1}} + \vec{\boldsymbol{b}}_{\boldsymbol{f}}\right) \tag{19}$$

where $\vec{\boldsymbol{f}}_t$ is the forget gate value ranging between 0 and 1, and other variables can be interpreted as previously indicated. The relationship between the current state of the LSTM cell, $\vec{\boldsymbol{C}}_{\boldsymbol{t}}$, and its past state, $\vec{\boldsymbol{C}}_{\boldsymbol{t-1}}$, the input gate value, $\vec{\boldsymbol{i}}_{\boldsymbol{t}}$, and the forget gate, $\vec{\boldsymbol{f}}_{\boldsymbol{t}}$, is shown in the following equation:

$$\vec{\boldsymbol{C}}_{\boldsymbol{t}} = \vec{\boldsymbol{i}}_{\boldsymbol{t}} \odot \hat{\vec{\boldsymbol{C}}}_{\boldsymbol{t}} + \vec{\boldsymbol{f}}_{\boldsymbol{t}} \odot \vec{\boldsymbol{C}}_{\boldsymbol{t-1}}, \tag{20}$$

where $\odot$ denotes element-wise multiplication. As can be seen, $\vec{\boldsymbol{i}}_{\boldsymbol{t}}$ is multiplied by $\hat{\vec{\boldsymbol{C}}}_{\boldsymbol{t}}$ to control how much input information can flow into the LSTM cell. Similarly, $\vec{\boldsymbol{f}}_{\boldsymbol{t}}$ is multiplied by $\vec{\boldsymbol{C}}_{\boldsymbol{t-1}}$ to control how much previous information in the cell is retained. How much $\hat{\vec{\boldsymbol{C}}}_{\boldsymbol{t}}$ can be outputted depends on the output gate, which is shown in the following:

$$\vec{\boldsymbol{o}}_{\boldsymbol{t}} = sigmoid\left(\boldsymbol{W}_o\vec{\boldsymbol{x}}_{\boldsymbol{t}} + \boldsymbol{U}_o\vec{\boldsymbol{h}}_{\boldsymbol{t-1}} + \vec{\boldsymbol{b}}_{\boldsymbol{o}}\right), \tag{21}$$

where $\vec{\boldsymbol{o}}_{\boldsymbol{t}}$ is the output gate value, and other variables can be interpreted as previously indicated. Finally, the output of the LSTM unit, $\vec{\boldsymbol{h}}_{\boldsymbol{t}}$, which is referred to as the *LSTM Hidden Unit Value* in this paper, can be obtained by the element-wise multiplication of the output gate value $\vec{\boldsymbol{o}}_{\boldsymbol{t}}$, and the hyperbolic tangent transformation of the LSTM cell's current state, $\vec{\boldsymbol{C}}_{\boldsymbol{t}}$, as follows:

$$\vec{\boldsymbol{h}}_{\boldsymbol{t}} = \vec{\boldsymbol{o}}_{\boldsymbol{t}} \odot tanh\left(\vec{\boldsymbol{C}}_{\boldsymbol{t}}\right). \tag{22}$$

Overall, input information first flows into the LSTM unit through a hyperbolic tangent activation function. Then, it passes through the input gate to enter the cell. Next, information in

the cell passes through the output gate to the sigmoid activation function, the value of which then flows into the output unit, y. The information in the cell also passes through the forget gate to influence the cell state of the next time point.

Similar to feedforward neural networks, LSTM parameter matrices are also estimated with maximum likelihood and optimized with some variants of the gradient descent algorithm (e.g., rmsprop and adadelta; see Goodfellow et al., 2016). The dropout algorithm is often used to prevent overfitting.

Applications of deep learning in educational data mining. Only very recently (i.e., after 2015) have researchers begun to apply deep learning in educational data mining. Currently, deep learning is the most commonly used algorithm to solve the problem of *knowledge tracing* (Piech et al., 2015; Huang & Brusilovsky, 2016; Khajah, Lindsey, & Mozer, 2016, Xiong et al. 2016; Huang & Brusilovsky, 2016). The goal of knowledge tracing is to predict whether a student can correctly solve the next test item, based on the test items the student has already solved in previous steps. More formally, the problem of knowledge tracing can be defined as: "given observations of interactions $x_0 \cdots x_t$ taken by a student on a particular learning task, predict aspects of their next interaction x_{t+1}" (Piech et al., 2015, p 2). Since the input of the knowledge tracing problem is sequential, LSTM is the most appropriate neural network modeling choice (Piech et al., 2015; Huang & Brusilovsky, 2016). Knowledge tracing can benefit e-learning by determining what test/practice item is the most appropriate for a student (i.e., the item's difficulty should be slightly higher than the student's ability) and determining the relationship between different concepts in a curriculum. Besides knowledge tracing, another area of deep learning application is *automated essay scoring*, or automated short answer grading (Zhang, Shah, & Chi, 2016). The goal of automated essay scoring is to train the computer to

automatically grade students' essays or short answer responses. Zhang, Shah, and Chi (2016) showed that deep learning algorithm (i.e., Deep Belief Network) outperformed other classification algorithms such as support vector machines and Naïve Bayes, and achieved high consistency with human raters. There are a few other novel applications of deep learning in educational data mining. For example, Sharma et al. (2016) used a deep convolutionary neural network to extract visual features from educational videos and used LSTM to combine visual and audio information to predict the liveliness of the videos. The results showed that this combined deep learning approach had significantly higher prediction accuracy than previous methods. Predicting the liveliness of educational videos can help e-learning programs to select and provide more engaging educational videos to students. Another novel application is to use LSTM to predict human tutors' verbal feedback based on students' log files and facial action units (Min et al., 2016). The result of the study can help the intelligent tutoring system to provide more dynamic and realistic feedback. Overall, LSTM is currently the most commonly used deep learning algorithm in educational data mining despite the fact that deep learning application in educational data mining is still relatively a new area of research.

Sequential Pattern Analysis in Educational Data Mining

Besides clustering and classification, sequential analysis is another major area in educational data mining (Zhou et al., 2011). The goal of sequential pattern analysis is to find common subsequence patterns in a set of sequences. For example, the log files generated by e-learning programs contain all students' problem solving sequences. Sequential pattern analysis can be used to discover the commonly used problem solving strategies by identifying the frequent subsequences in the log files. Sequential pattern analysis has been applied in educational data mining in two ways, including: helping to customize the e-learning environment

based on students' activity patterns (Zaïane, & Luo, 2001), and identifying problem solving patterns that are markers of success (Kay, Maisonneuve, Yacef, & Zaiane, 2006). In the following subsections, sequential pattern analysis will be formally introduced, then the three stages of applying sequential pattern analysis in the analysis of log files will be discussed, and finally, studies that apply sequential pattern analysis in educational data mining will be reviewed.

An introduction to sequential pattern analysis. The concept of sequential pattern analysis was first proposed by Agrawal and Srikant (1995). To formally introduce sequential pattern analysis, some basic terminology is required. First, a *sequence* is defined as an ordering of events and each event in the sequence is called an *item*. A sequence, α, is the *subsequence* of another sequence, β, if α can be formed from β by removing some items in β but without changing the relative positions of the remaining items in β. If α is a subsequence of β, we can also say, β contains α. For example, if $\alpha = \{A, B, C, D\}$ and $\beta = \{A, D, B, E, C, D\}$, then α is a subsequence of β because α can be formed by removing the first D and E in β. If $\gamma = \{A, B, D, C, E\}$, then α is not a subsequence of γ because there is no way to obtain α from γ without changing the order of the items in γ. Continuing with this basic terminology, given a set of sequences S, the *support* of a sequence α is defined as the number of sequences in S that contain α. Sequence α is a *sequential pattern* if α's support is greater than the *minimum support threshold* set by researchers. For example, if a researcher sets the minimum support threshold to 8, then there must be at least 8 sequences that contain α, in order for α to be considered a sequential pattern. Finally, the goal of sequential pattern analysis is to find all sequential patterns in set S. Various computer algorithms were developed to iteratively search for sequential patterns in a set. Zhou et al. (2011) reviewed various sequential pattern search algorithms and suggested *PrefixSpan* (Pei et al., 2001) as the currently most efficient searching algorithm. Again, the

optimization details are not the focus of this paper. Interested readers can check the original paper by Pei et al. (2001). Note that all the searching algorithms return the same results; they only differ in computational speeds.

Three steps of applying sequential pattern analysis. Zhou et al. (2011) illustrated a three-step approach to apply sequential pattern analysis to mine log data. The three-steps are preprocessing, pattern discovery, and pattern analysis. In the first step, log data need to be preprocessed because it contains too many problem-irrelevant actions, which can complicate the sequential pattern analysis. Thus, researchers need to create an *action library* that specifies all the action items that are of theoretical interest. According to Zhou et al. (2011), an *action* in an action library consists of a sequence of events from the raw log files. Researchers can group specific sequences of events in the raw log files into higher order actions based on theory. This allows researchers to change the grain size of sequential pattern analysis, and it simplifies the analysis complexity. Sometimes, it is necessary to compile a sequence of actions into a higher-order action. For example, in order to add two-digit numbers, students may perform a sequence of actions. This sequence of actions can be combined into one higher-order action, namely, addition. This practice is done to prevent sequential pattern analysis from producing too many sequential patterns that cannot be meaningfully interpreted by researchers. Researchers need to write simple computer programs to compile the action library efficiently. An example code can be found in Zhou, Xu, Nesbit, and Winne's (2011) book chapter. After the action library is compiled, the rest of the events in the log file can be eliminated to simplify the data mining process.

In the second step, researchers can apply sequential pattern mining algorithms to the sequences of actions created from the first step. In order to discover meaningful patterns,

researchers often need to add several constraints based on their research questions or prior knowledge. For example, if the research question is about strategies students adopt in the revision phase of essay writing, then researchers can constrain the sequential pattern analysis to extract only sequences within the last 15 minutes of the log files. Another example is when researchers are interested in identifying the antecedent and subsequent actions of a target action. In that case, researchers can constrain the sequential pattern analysis to extract sequences only within a certain distance from the target action. Besides adding constraints, researchers can also divide students into groups based on other psychological/demographic variables, and then conduct sequential pattern analysis within each group.

In the third step, the goal is to interpret the meaning of the patterns discovered in step two and then confirm the validity of the previously discovered patterns by conducting sequential pattern analysis and other statistical analysis using a new dataset. According to Zhou et al. (2011), step 2 may yield thousands of frequent patterns. In order to interpret which patterns are meaningful, researchers can filter the sequential patterns based on length, frequency, and particular actions contained in a pattern. Researchers need to rely partly on theoretical assumptions to complete this step. After researchers identify a set of meaningful patterns, researchers can conduct sequential pattern analysis again in a testing dataset to check if the previously identified patterns can be identified again. Researchers can also study the relationship between the identified patterns and other outcome variables to see which patterns can significantly predict the outcome variables.

Applications of sequential pattern analysis in educational data mining. In the educational data mining literature, sequential pattern analysis is often used as a feature extraction tool to improve classification (e.g., Kay, Maisonneuve, Yacef, & Zaiane, 2006; Romero, Ventura,

Zafra, & de Bra, 2009; Wang, Weng, Su, & Tseng, 2004). Usually, the procedure is to simplify the log data by constructing an action library, using sequential pattern analysis to extract patterns, creating binary pattern indicator variables that denote whether a pattern is present in each student's problem solving sequence, and inputting these pattern indicator variables to a classification algorithm to predict outcome variables. The outcome variables could be activities/resources that best suit a student in terms of difficulty and preference (Romero, Ventura, Zafra, & de Bra, 2009; Wang, Weng, Su, & Tseng, 2004), or it could be the success in solving a problem (Kay et al., 2006; Luan, 2002). Besides improving prediction, sequential pattern analysis can also be used to evaluate whether students' action sequences match with a desired or expert sequence (Pahl & Donnellan, 2003). Currently, the two major limitations of sequential pattern analysis include reliance on researchers' prior knowledge to identify relevant actions and the inability to extract less frequent but important patterns (e.g., a novel problem solving strategy used by a small group of creative students).

Research Problem

The central problem that this study attempts to solve is how to analyze log data of students' problem-solving sequences in simulation-based assessments. There are many challenges to solving this problem. First, it is important to consider the order of student actions. To illustrate this, consider the following simple math problem, $2 + 3 \times 4$. Solving this problem by doing addition before multiplication will produce very different result than doing multiplication before addition. Second, many actions performed by students and recorded in the log files may be irrelevant to the problem (e.g., students can play around in a simulation-based assessment environment without really attempting to solve the problem), and there is currently no formal theory that accounts for relevance of actions. Third, a student can change and try

several strategies during the problem-solving process. The data mining methods in the literature, if used alone, are insufficient to address these challenges. Cluster analysis ignores student action order and relies on researchers' subjective knowledge to determine which actions in log files are relevant. While some classification techniques such as LSTM do take student action order into account and can determine the relevance of the log file actions, they cannot cluster the sequences. Sequential pattern analysis does consider the order of the sequences but this procedure relies on researchers' knowledge to rule out problem irrelevant actions. In addition, sequential pattern analysis may not be able to detect less frequent but nonetheless important problem-solving patterns (e.g., creative problem-solving strategies used by a few students). One solution is to integrate the three methods so they can complement each other in order to achieve the best result. More specifically, the problem of subjective feature selection in cluster analysis and sequential pattern analysis can be addressed by first extracting problem-relevant features using LSTM. That is, when an LSTM network is trained to predict the problem-solving outcome using problem-solving sequences, the LSTM network extracts important features from the input sequences and stores these features as hidden unit values (i.e., cell values in an LSTM unit; LeCun, Bengio, & Hinton, 2015). In contrast to the subjective feature selection based on expert reviews, this procedure represents a data driven approach to extracting important features that best predict student problem solving outcomes. Then, cluster analysis can be applied to the statistically identified features, stored in the hidden units in the LSTM, to group students into different categories. Finally, sequential pattern analysis can be applied to interpret the meaning of each cluster. The detailed procedure of the proposed method is presented in the next chapter.

Chapter 3 Proposed Method: LSTM Cluster

The goal of the proposed method is to extract problem-relevant features from log files using LSTM, cluster the sequences based on LSTM extracted features, and interpret each cluster by examining exemplar sequences that have outcome probabilities close to the center of each cluster. Therefore, the proposed method has three main steps. Each of these steps is discussed next.

LSTM Classification

The goal of this step is to extract problem relevant features from the input sequences. That is, when an LSTM network is trained to predict problem solving outcomes using problem-solving sequences, the LSTM network extracts important nonlinear features from the input sequences and later uses these features (i.e., hidden unit values, $\vec{h}_t$ in Equation 22, LeCun, Bengio, & Hinton, 2015) to predict the outcome. The proposed LSTM model used in the present study is shown in Figure 7. As can be seen, the major difference between Figures 6 and 7 is that Figure 7 has output units only at the end of the sequence (instead of at each time point). This design is to allow the network to process the entire problem-solving sequence (i.e., the input) before making a prediction about whether a student can successfully solve the problem (i.e., the output). To illustrate this with an example, a student may conduct a series of experiments (i.e., the problem-solving sequences) in a simulation-based assessment to reach a final conclusion about the relationship between the payload mass of a balloon and the altitude at which the balloon can fly; the latter is the problem-solving outcome, which can be scored as correct or incorrect. Some of the actions the student performs are relevant to the problem (e.g., constructing graph and tables), but others are trivial (e.g., random clicking). Also, the order of the actions can be important. The LSTM network processes all the actions in a sequence, and extracts important

non-linear features from these actions, before making a final prediction of the outcome. Assuming the problem-solving outcome is a binary variable, the output unit's activation function can be set to be the sigmoid function, which outputs the probability of correctly solving the problem. This probability is referred to as the *LSTM output unit probability* in this paper. All the equations associated with the LSTM units are the same as discussed in the previous section. In practice, researchers need to determine the number of cells within an LSTM unit. This decision depends on the complexity of the problem to be solved by the network. In general, the number of cells can be set equal to the dimension of the input vector (Carrier & Cho, 2016). The idea is to include as many cell values in an LSTM unit as possible, and then use the weight decay method to eliminate extra cells (Goodfellow et al., 2016).

Before training the network, students' problem-solving sequences need to be recoded. Each action is coded using *one hot coding*, which is similar to dummy coding in statistics. For example, suppose there are three total possible actions. Action 1 can be coded as (1 0 0), action 2 can be coded as (0 1 0), and action 3 can be coded as (0 0 1). That is, if there are three total possible actions, there will be three input variables. If there are too many actions (e.g., thousands of unique actions), then *embedding* can be used to recode the hot coding (Goodfellow et al., 2016). Embedding is a technique that transforms binary one hot coding into continuous coding with smaller dimensionality. After recoding the raw data, the ML method can be used to estimate the parameters. A variation of the gradient descent algorithm, the Adadelta algorithm (Zeiler, 2012), can be used to find the ML solution. It is also recommended to use the weight decay regularization and dropout algorithm to prevent overfitting. For a detailed explanation of these optimization techniques, see Goodfellow et al. (2016). To train the LSTM, data are divided into three datasets: training dataset, validation dataset, and testing dataset. The training dataset is

used to estimate the parameters. The validation dataset is used to assess whether the model is overfitting the data. The model estimated in the training dataset is used to predict the testing dataset. If the model has low error rate in training dataset, but high error rate in validation dataset, then it indicates that the model is not generalizable. Thus, to avoid the problem of overfitting, a model with the lowest validation error rate is selected as the final model. This final model is then used to predict the test dataset. After selecting the best model, hidden unit values in LSTM are saved and then analyzed in the next step.

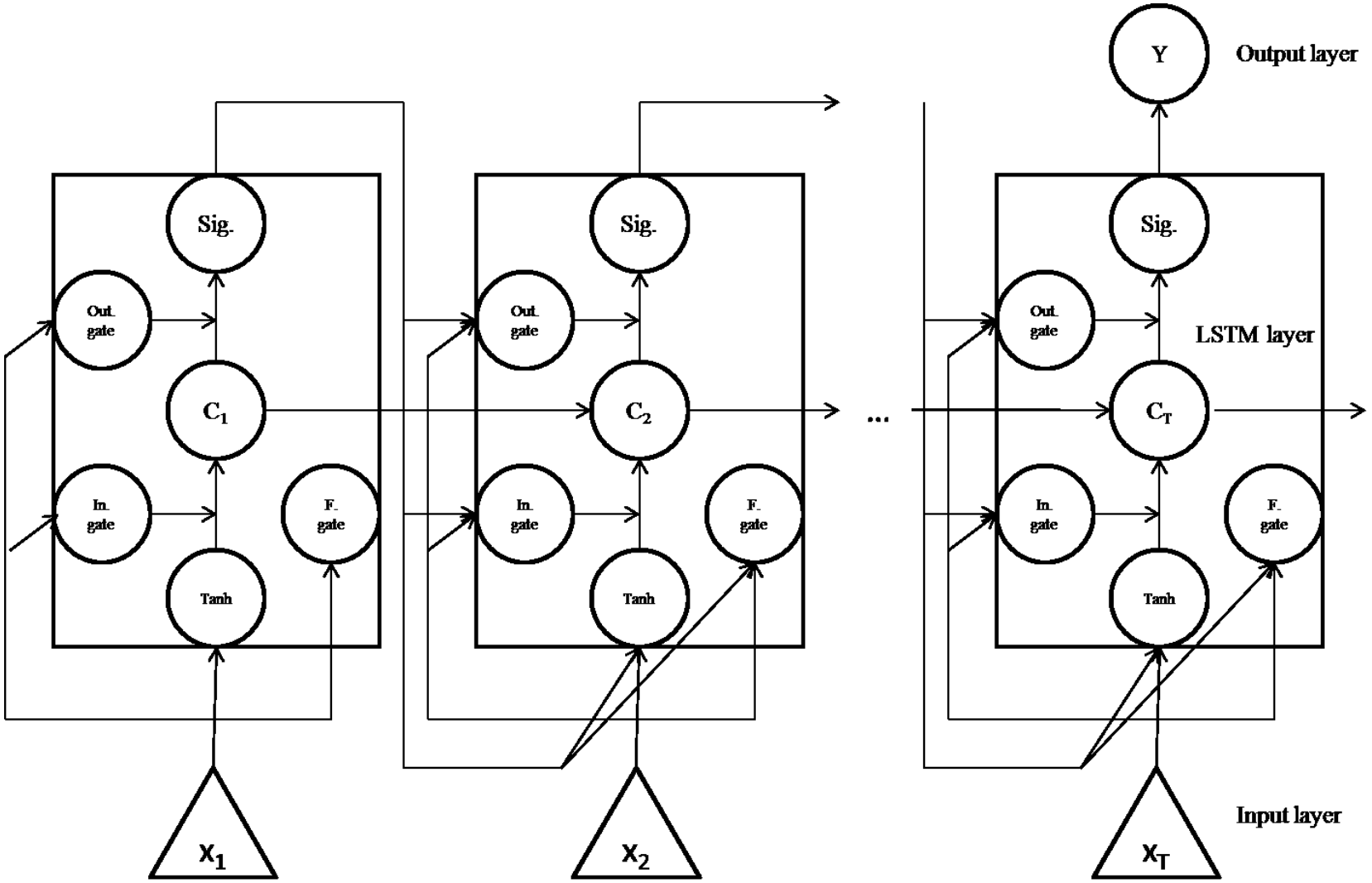

Figure 7. A conceptual diagram for a LSTM network unfolded over time. X = input, Y = output, Sig.=sigmoid function, Tanh = hyperbolic tangent function, Out. gate = output gate, In. gate = input gate, F. gate = forget gate, C = cell, T = sequence size.

Cluster Analysis

In this step, different cluster analyses techniques can be applied to cluster students based on LSTM's hidden unit values. The rationale for this step is to use LSTM extracted features (i.e., LSTM hidden unit values) from student problem-solving steps as cluster analysis inputs to create more meaningful clusters of students' problem-solving strategies. Since LSTM's hidden unit values are continuous variables, *k*-mean cluster analysis or Gaussian mixture model (Vellido et al., 2011) can be used. To determine the number of clusters, different statistical tests as well as theoretical interpretation may need to be considered. The challenge here is that different clusters of student problem-solving strategies tend to form higher-order clusters. For example, all the correct strategies may form one cluster and all the incorrect strategies may form another cluster. This could make many of the cluster evaluation statistics less accurate. To illustrate the impact of such a data structure on different cluster evaluation statistics, consider the following: a two dimensional dataset with five clusters nested within three higher-order clusters. The data set is visualized using scatterplot in Figure 8.

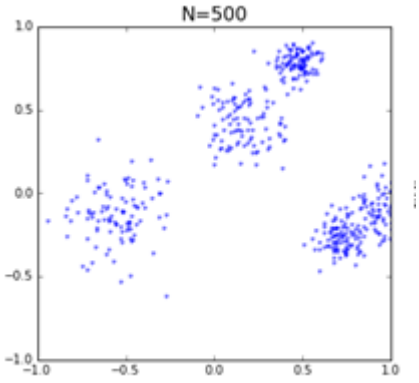

Figure 8. Scatterplot of simulated dataset with five clusters nested in three bigger clusters. Different evaluation statistics were run on this dataset. AIC, BIC, *f(k)*, and bootstrap likelihood ratio test identified three clusters, while gap statistics identified only one cluster. However, by

examining the scatterplot visually, it is not hard to see that there are four or five clusters. This example demonstrates the advantage of applying a graphical method over current statistical methods when cluster structure is nested. That being said, LSTM hidden unit values often have more than two dimensions. To use the visualization method, the dimensions of LSTM hidden unit values need to be reduced. It is important to note that the LSTM output unit automatically combines LSTM hidden unit values into one dimension, the LSTM output unit probability. While combining LSTM hidden unit values, regularization decay weights are applied. As discussed earlier, decay weights can eliminate hidden units that are not important for predicting the outcome. This property makes LSTM output unit probability more desirable than other methods such as PCA discussed in the literature review. Therefore, I propose to use an LSTM output unit probability histogram to help visually determine the number of clusters. However, it is important to note that when we reduce the dimensions of the LSTM hidden unit values, information loss is inevitable. If two different problem-solving strategies lead to the same probability of correctly answering the question, this method would lead to an underestimation of the number of clusters. In practice, it is better to determine the number of clusters based on several criteria as well as the meaningfulness of interpretation.

Interpretation of the Clusters

Due to the sequential nature of the log files, the meaning of each cluster cannot be determined simply by looking at action frequency statistics. In this study, I propose to use exemplar sequences and sequential patterns to represent and interpret the meaning of each cluster. For exemplar sequences, the basic idea is rooted in the *Prototype* (Rosch, 1975) and *Exemplar* theories (Smith & Medin, 1999) of concepts in cognitive psychology. The prototype theory suggests that people represent a concept using an average of all instances that belong to that

concept. In contrast, exemplar theory suggests that a concept is represented by the most frequent instances of that category. Regardless of which theory is most appropriate, we can borrow the idea that a cluster/concept can be represented by a few of the most representative instances. In statistical terms, we can identify problem-solving sequences that have LSTM hidden unit values closest to the centroid of each cluster and use these sequences to represent the cluster. These sequences are thus named as the exemplar sequences.

Sequential patterns for each cluster can be obtained from sequential pattern analysis discussed in the literature review section. Since a sequential pattern is a subsequence of the original sequences, direct interpretation can be difficult. However, sequential patterns can be useful at checking the representativeness of the exemplar sequences. Ideally, exemplar sequences should contain all sequential patterns. If exemplar sequences do not contain all sequential patterns, then it is necessary to identify other exemplar sequences that contain the rest of the sequential patterns. This is to ensure the exemplar sequences do not miss any frequent patterns in the corresponding cluster by chance. To do this, sequences within a cluster are sorted based on their distance from the centroid. Sequences that are closet to the centroid and contain the missing sequential patterns are chosen. Then exemplar sequences are inspected to identify the strategy used to solve the problem and its strengths and weaknesses in problem solving.

In order to demonstrate the procedure of the LSTM cluster, I applied the proposed method to log files of students solving a series of problems in a National Assessment of Educational Progress (NAEP) simulation based assessment, the *Technology Rich Environment* (TRE) Science Laboratory (Bennett, Persky, Weiss, & Jenkins, 2007). A simulation study was also conducted to evaluate the accuracy of the LSTM cluster. The methods and results of the real data study and the simulation study are presented in Chapters 4 and 5, respectively.

Chapter 4 Analysis of Real log files with the LSTM Cluster

In this chapter, LSTM is demonstrated with a real dataset from Chu's (2016) study. Before LSTM cluster was applied to the dataset, a review of the real dataset was conducted to explore whether LSTM cluster was suitable for analyzing this dataset. The major goal of this chapter was to examine whether the LSTM cluster results could be meaningfully interpreted. More specifically, the following research questions were addressed:

1) What student problem-solving strategies were identified by reviewing the real data?
2) Did review of the real data suggest that the orders of student actions were important in solving the problem?
3) Can LSTM accurately predict the training, validating, and test data sets?
4) How many clusters of problem-solving strategies were identified using LSTM cluster?
5) Can the clusters be meaningfully interpreted using exemplar sequences and sequential pattern analysis?
6) Did LSTM cluster and qualitative review identify the same strategies? If not, what are the differences?

Method

Participants. The data used in this study were initially collected by Chu (2016) where 298 8th-grade science students from Alberta, Canada were asked to solve problems in a National Assessment of Educational Progress (NAEP) simulation based assessment, the *Technology Rich Environment* (TRE) Science Laboratory (Bennett, Persky, Weiss, & Jenkins, 2007). Out of these 298 students, 30 students did not provide consent to use their log files, and 75 students encountered technical difficulty during the experiment. Thus, the data from only 193 students were kept in the final analysis. Out of these 193 students, 49% were male, 46% were female, 5%

did not disclose their gender. The largest ethnicity group was Caucasian (38%) followed by Filipino (24%), African American (10%), Latin American (10%), East Asian (4%), Southeast Asian (3%), South Asian (3%), and others (4%). Majority of the students (97%) reported that they used computer at home.

Description of the TRE science laboratory problem. In the TRE science laboratory, students were asked to design and conduct a series of three experiments to deduce the relationship between payload mass, the amount of starting helium, and the altitude of a balloon. Note that although balloon science is not explicitly covered in the Alberta grade 8 Science curriculum, the contents were considered conceptually related to the local science program (Chu, 2016). Thus, although it was expected that the students did not have any explicit prior knowledge about balloon science, they had the general science knowledge to design experiments to deduce the relationship.

More specifically, in the TRE science laboratory, students were first told that they would use a simulation tool to solve a set of problems related to balloon science. Then, a brief introduction of the simulation as a scientific tool was provided. Next, a tutorial of the computer based simulation tool was provided. The tutorial explained the location of the research questions, how to design simulation experiments by selecting different amount of payload mass, how to make predictions before conducting the simulation, how to run the experiments and observe the results, and how to construct tables and graphs to visualize the results. After the tutorial, the first research question was presented to students: “How do different payload masses affect the altitude of a helium balloon?” In order to solve this problem (assuming students had no prior knowledge about this problem), students need to correctly identify the independent variable (IV; payload mass) and dependent variable (DV; altitude of the balloon), conduct at least three

experiments by choosing three numbers of payload masses ranging from 10 lbs. to 90 lbs., run these experiments, construct a table or graph with variable payload masses as the independent variable and altitude as the dependent variable, and finally answer the multiple-choice question in the conclusion section. The multiple-choice question at the end of the assessment is presented below, and b is the correct answer.

Based on your experiments, which statement most accurately and completely describes how different payload masses affect balloon altitude?

a) *The less the payload mass, the lower the altitude reached by the balloon.*
b) *The greater the payload mass, the lower the altitude reached by the balloon. (Key)*
c) *Changing the payload mass does not change balloon altitude.*
d) *Increasing payload mass increases balloon volume and therefore balloon altitude.*
e) *The greater the payload mass, the greater the altitude reached by the balloon for any amount of helium.*
f) *The less the payload mass, the greater the altitude reached by the balloon for any amount of helium.*

A snapshot of the TRE simulation software is presented in Figure 9. In total, there are 22 relevant actions in the log files and their meanings are presented in Table 2. Note that action glossary, science help, and computer help were not included in Table 2 because few students used these options. Some students clicked these buttons, but they immediately exited the help screen without really clicking any subtopics. Hypothesis related actions were also not included due to majority of the students choosing the “I don’t know” option when making a hypothesis. After solving problem 1, there were also problems 2 and 3. Since the purpose of the analysis was

to demonstrate the proposed method, problems 2 and 3 were not the focus of the study and will not be further discussed.

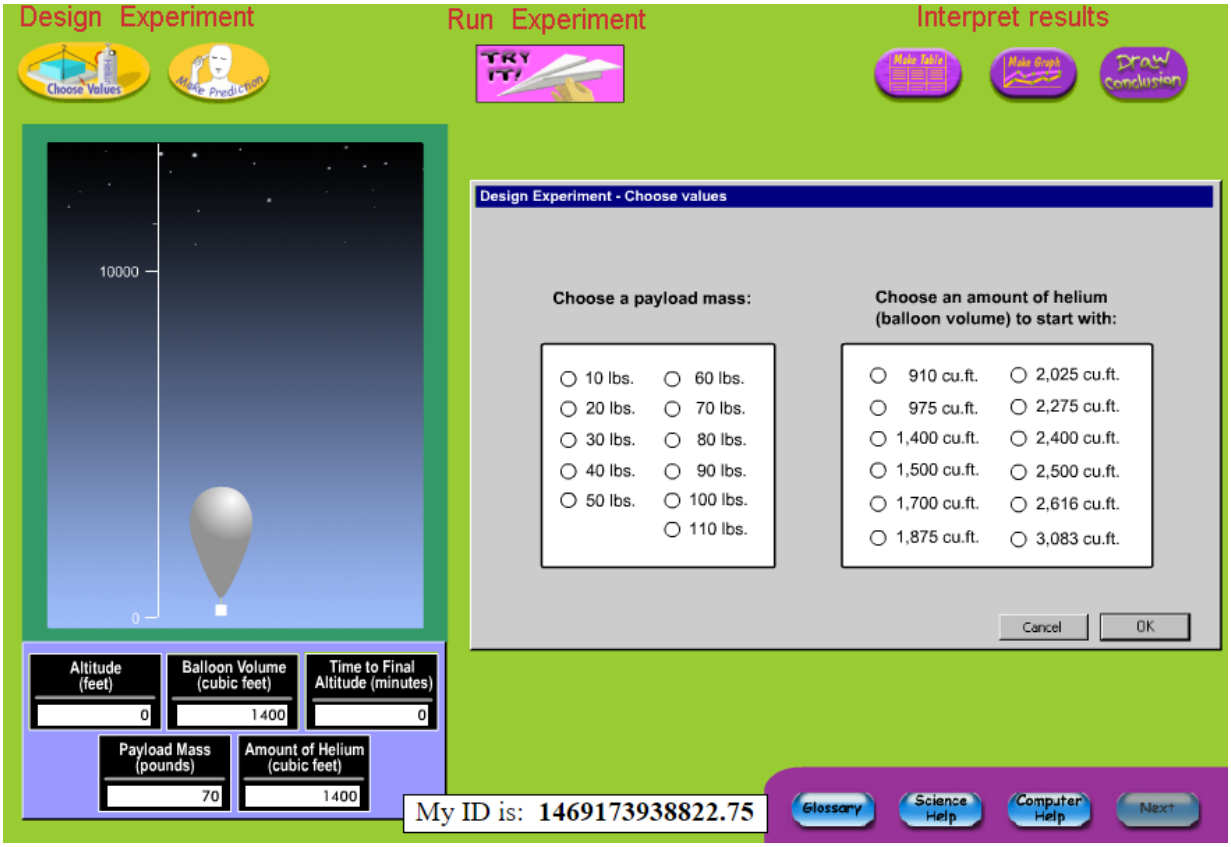

Figure 9. A snapshot of the TRE science laboratory.

Table 2. *Actions from TRE simulation log files.*

Action name in log file	Meaning
Begin_problem1	Students click this button to begin problem 1
selectedMass_10	Students click this button to set the payload mass of the balloon to be 10 lbs
selectedMass_20	Similar as above
selectedMass_30	Similar as above
selectedMass_40	Similar as above
selectedMass_50	Similar as above
selectedMass_60	Similar as above
selectedMass_70	Similar as above
selectedMass_80	Similar as above
selectedMass_90	Similar as above
tryIt	Students click this button to run the simulation and observe how high the balloon can fly

makeGraph	Students click this button to start generating a graph
vertAxis_altitude	Select altitude as the y variable of the graph
horizAxis_mass	Select payload mass the x variable of the graph
horizAxis_helium	Select helium volume as the x variable of the graph
vertAxis_time	Select time to final altitude as the y variable of the graph
Make_Table	Initialize the table
selectedTableVars_Altitude_(ft.)	Select altitude as one of the table variable
selectedTableVars_Payload_Mass_(lbs.)	Similar as above
selectedTableVars_Amount_of_Helium(cu.ft.)	Similar as above
selectedTableVars_Time to Final_Altitude_(min.)	Similar as above
Problem1Question2	Initialize the multiple choice item in the end

Analysis. Before applying LSTM cluster to student log files, it was helpful to explore student problem-solving sequences and strategies by conducting a review of the log files. For example, to understand students' problem-solving strategies, frequent sequence patterns were identified (i.e., just like identifying themes from qualitative data). While the results from this review could be considered informal, they provide some insights into whether the orders of student actions were indeed important in solving the TRE science laboratory problem, which helped justify the use of LSTM cluster in analyzing the data and assisted in the interpretation of the clustering results. Additionally, this review also provided a useful basis for the following simulation study.

After reviewing the log files, LSTM cluster was applied to analyze the data. Since the sample size was relatively small for a rather complex statistical model such as the LSTM, I decided to reduce the complexity of the log files by removing irrelevant actions and grouping theoretically or procedurally related actions. More specifically, all selectedMass_X actions in Table 2 were combined with the tryIt action. That is, if any selectedMass_X action was followed

by the tryIt action, then the two actions were combined to form the new *experiment* action. It was also assumed that the actual quantity of mass a student selected did not matter very much as long as student ran multiple unique experiments. For example, both {selectedMass_10, tryIt} and {selectedMass_30, tryIt} were considered to be the experiment action. Next, the makeGraph action in Table 2 was grouped with other select-graph-variable actions (i.e., vertAxis_altitude, horizAxis_mass, horizAxis_helium, & vertAxis_time). If the combination included makeGraph, vertAxis_altitude, and horizAxis_mass, then the combined action was labeled correct graph. Otherwise, the combined action was labelled wrong graph. Finally, table-related actions were combined in a similar way. If the combined table actions included Make_table, selectedTableVarselectedTableVars_Altitude_(ft.), and selectedTableVars_Payload_Mass_(lbs.), then the combined action was labelled as correct table. Otherwise, it was labelled wrong table. The other actions in Table 2 were removed from the sequences. After data preprocessing, each student's problem-solving sequence contained only five possible actions (i.e., experiment, correct graph, wrong graph, correct table, and wrong table), although the five actions could be repeated and placed in different orders.

Then, the LSTM cluster was applied to analyze the cleaned log data. In the first step, LSTM was applied to model the data, which was randomly divided into three sub datasets: training dataset (n=135), validation dataset (n=29), and testing dataset (n=29). The LSTM neural network (programmed using the *theano* package in Python 2.7; available upon request from the author) had five input units, three cells, and one output unit. The number of hidden units (cells) was experimentally determined. The number of hidden units that led to the lowest error rate in the validation dataset was chosen. For example, in order to determine the number of hidden units, the dataset was split into three sub-datasets: training, validation, and testing datasets. Next,

LSTM networks with 2 to 20 hidden units were trained, and their prediction error rates in validation dataset were recorded. Since the validation dataset error rates flattened after applying the 3 hidden unit model, the number of hidden units was set to 3. Adadelta and drop out algorithms were used to optimize the neural network weights. The regularization decay factor was set to be .01. Maximum number of iterations to convergence was set to 1000.

In the second step of LSTM cluster, cluster analysis of the LSTM hidden values was conducted using three cluster analysis algorithms: *k*-mean, Gaussian mixture model, and agglomerative cluster analysis. *K*-mean cluster analysis was conducted using the *scikit-learn* package in python 2.7 (Pedregosa et al., 2011). The number of clusters was set based on the number of peaks in the LSTM output unit probability histogram. Gaussian mixture model or latent profile analysis was conducted using Mplus 7 (Muthén & Muthén, 1998-2015). Agglomerative cluster analysis with average link distance (between-groups linkage in SPSS) was conducted using SPSS version 24. The ML estimation method was used to estimate the model parameters. To ensure global maximum was obtained, 2000 sets of start values in the first step of optimization were used. In the second step of optimization, the 50 start value sets that showed the largest log likelihood values in the first step were picked and optimized until convergence was achieved (Geiser, 2010). Bootstrap log ratio difference test was used to test the relative model fit and determine the number of clusters. Five hundred bootstrap samples were used based on the recommendation of Geiser (2010).

In the third step of LSTM cluster, sequential pattern analysis was conducted using the pymining package (Dagenais, 2015) in Python 2.7. The minimum support threshold was set to be 50% of the cluster size. For example, if a cluster has 10 cases, then the minimum support threshold is set to 5. Redundant subsequences were removed from the final results. That is, if

sequence ABC and AB both met the minimum support threshold, sequence AB (which is a subsequence of ABC) will be removed. These decisions are made to reduce the number of sequential patterns, so that it is easier to interpret them.

Results and Discussion

Review of the log files.

RQ1. What student problem-solving strategies were identified by review of the real data?

Through the review of the log files, five major problem-solving strategies were identified. The first type of strategy was labelled as the *Table Strategy*. Students who used the table strategy tended to conduct more than 5 experiments and rely on tables for solving the problem. They may or may not also produce graphs. When they produced a graph, they could not correctly identify the right variables. The common feature of students who used the table strategy was that they produced a table with all five variables. The correct table was produced either at the beginning of their attempts or after several failed attempts. The second strategy that was identified was labeled as the *Graph Strategy*. Students who used the graph strategy conducted at least three experiments and solved the problem using graphs. The most defining feature of this group of students was that they could produce a graph with the correct IV and DV. Most of these students did not use the table at all. The third strategy was labeled as the *"Try-Them-All" Strategy*. Students often conducted many experiments (more than 5), and they typically produced both a graph with the correct IV and DV and a table with all the variables. The fourth strategy was labeled as the *Random Try Strategy*. Students often conducted more than five experiments, and they would try to produce the table or graph but with incorrect variables. The fifth strategy was labeled as the *Guess Strategy*. Students often conducted a very few number of experiments (less than three),

and sometimes produced the correct table or graph. The most apparent feature of the guess strategy was the short lengths of the log files.

RQ2. Did the review of the real data suggest that the orders of actions were indeed important in solving the problem?

The review of the log data suggested that the orders of actions were important in solving the problem, but at the same time, revealed variability or flexibility in allowing for different paths to solutions. This means that unlike solving a well-defined math problem, in which there was one (or few) fixed order of actions to solve it, there was no fixed order of actions to solve the TRE science laboratory problem. Students could perform different actions in different orders to solve the problem. In the TRE science laboratory problem, orders of actions mattered in a subtler way. For example, when students made predictions about the outcome of the current experiment, they needed to make a judgment about whether the balloon would rise to a higher, lower or equal altitude than in the previous experiment. The correctness of the prediction depended on the payload mass students selected in the previous experiment as well as the payload mass they selected in the current experiment. Another example would be related to students' learning process. Students in one group failed to make the correct tables/graphs in the initial attempts, but after they conducted a series of experiments, they correctly created the tables/graphs. Consequently, they could correctly answer the question. Students in a second group created the correct tables/graphs initially in their attempts, but then they started to generate the wrong tables/graphs before they conducted enough experiments. As a result, they failed to correctly answer the question. Yet, another group of students created the correct tables/graphs initially in their attempts. But after conducting several experiments, they began to explore other variables not directly related to the research questions by creating tables/graphs that were not

correct or unrelated to the research question. Since they conducted enough experiments with the correct tables/graphs, they could still correctly answer the question. The three groups of students performed similar actions, but in different orders which led to different outcomes. Thus, the review of log files suggested that in order to understand students' problem-solving behaviors in the TRE science laboratory environment, it is important to consider the order of actions, and sequential analysis techniques such as the LSTM are needed to analyze the problem.

LSTM cluster.

RQ3. Can LSTM accurately predict the training, validating, and testing the data set?

The LSTM network training took 45.1 seconds. The prediction error rates for training, validation, and testing datasets were 19%, 10%, and 14%, respectively. This result may seem unusual as, in general, training error rates tend to be smaller than validation and testing error rates. This unexpected result is likely due to the small sample sizes of the validation and testing datasets. It is expected that with larger validation and testing sample sizes, validation and testing error rates will approach the training error rates. However, since the validation and testing error rates were relatively low and consistent with each other, it provides some evidences for the validity of these estimates.

RQ4. How many clusters of problem-solving strategies were identified using LSTM cluster?

To determine the number of clusters, LSTM output unit probability histogram, AIC, BIC, and bootstrap LR test were used. The output unit probability histogram is shown in Figure 10. As can be seen, the output probability histogram seemed to be a mixture of four unique distributions. The distribution with the lowest mean probability had a peak at 0.17. The second, smaller distribution had a peak at 0.29. The third major distribution peaked around .54. The fourth

distribution centered around .72. AIC, BIC, and bootstrap LR test also provided evidence for a four-cluster solution as the values for AIC and BIC for four-cluster solutions (AIC=-318.55, BIC=-292.448) were lower than for the three-cluster solutions (AIC=-288.193, BIC=-268.617). Moreover, the bootstrap LR test showed that a four-cluster model fitted the data significantly better than three-cluster model ($p<.001$). However, the five--cluster Gaussian mixture model led to convergence issues. Consequently, AIC, BIC, and bootstrap LR tests could not be computed.

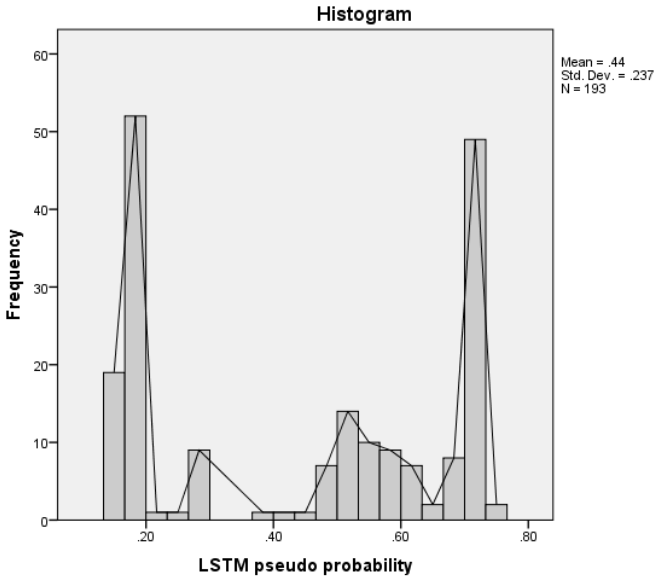

Figure 10. LSTM output unit probability histogram for real data.

Since both the histogram and the Gaussian mixture model suggested that there were four clusters of student strategies, a four-cluster *k*-mean analysis was also conducted on the LSTM's hidden unit values. The resulting cluster membership was similar to the Gaussian mixture model cluster membership, with only 4 cases being assigned into different clusters. The *k*-means cluster membership was also very similar to agglomerative cluster membership, with only 3 cases being assigned into different clusters. Since *k*-mean cluster analysis is a simpler algorithm with less convergence issues than Gaussian mixture model with faster computational speed than agglomerative cluster analysis, I will focus on the *k*-mean cluster analysis results in the following

sections. The mean probabilities to correctly answer the multiple-choice question for each cluster and number of cases in each cluster are presented in Table 3.

Table 3. *Cluster descriptive statistics*.

Cluster	Probability to correctly answer multiple-choice questions	Number of cases in each cluster
1	.29	11
2	.17	72
3	.54	48
4	.72	62

RQ5. Can the cluster be meaningfully interpreted using exemplar sequences and sequential pattern analysis?

In order to interpret the clusters, three exemplar sequences and all the sequential patterns for each cluster are shown in Table 4. Exemplar sequences are sequences with LSTM hidden values close to the center of each cluster. Cluster centers here can be represented using the cluster centroids. As shown in Table 4, cluster 1's exemplar sequences consisted of really short problem-solving sequences. Students in cluster 1 conducted very few (less than 3) experiments before they proceeded to guess the answer. The sequential pattern extracted was similar as the exemplar sequences. Overall, they had a mean probability of 0.29 to correctly answer the multiple-choice question. Thus, we labelled cluster 1 as the *Guess Strategy*.

In contrast to cluster 1, the exemplar sequences in cluster 2 showed that these students conducted significantly more experiments than students in cluster 1. However, students in cluster 2 had an even lower probability in correctly answering the final question (.17). A detailed examination of the cluster 2 sequences suggested that although these students conducted more experiments, they failed to make the correct table or graph. They produced tables/graphs using

the wrong variables, which led them to produce the wrong answer. This is probably why they had an even lower probability in answering the question correctly than students in cluster 1. It is important to note that in one exemplar sequence ([0, 2, 3, 4, 0, 0, 0], which means that the student first did an experiment (coded as 0), then constructed an incorrect graph (coded as 2), next constructed a correct table (coded as 3), followed by an incorrect table (coded as 4), finally completed the sequence by three experiments (0)), the student seemed to make the correct table in the beginning, but then immediately changed the correct table to the wrong table before conducting further experiments. Consequently, the student failed to answer the question correctly. If the student had produced the correct table at the end of the sequence, the student may have solved the problem. In this case, LSTM correctly used the order of action to predict the outcome probability. All of cluster 2's sequential patterns were subsequences of the exemplar sequences, indicating that the exemplar sequences were indeed representative for cluster 2. Since students in cluster 2 appeared to persevere but could not produce the correct table or graph, cluster 2 was labeled as the *Random Try Strategy*.

In cluster 3, the exemplar sequences (as well as the rest of the sequences) suggested that students in this group conducted two or more experiments, along with producing the correct table or graph. However, for some reason, after conducting a number of experiments and constructing the correct tables or graphs, these students decided to make the wrong table or graph at the end of the sequence. This could mean that after figuring out the problem with a sufficient number of experiments and the correct table or graph, the students decided to try the wrong graph out of curiosity; or it could mean that these students were confused but their attempts allowed them to deduce the correct answer anyway. As a result, students in cluster 3 had a moderate probability of correctly answering the question (.54). All the sequential patterns except for one, [0,1], were

subsequences of the exemplar sequences, indicating the representativeness of the exemplar sequences. The sequential pattern [0,1] indicated that students in cluster 3 also were able to construct the correct graph after an experiment. Some example sequences including [0,1] were: [0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 2], and [0, 0, 4, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 4, 4, 3]. Note that these students constructed the correct graphs (coded as 1_ in the beginning of the sequences, after a number of experiments, but they also constructed some incorrect graphs and/or tables near the end. This pattern was consistent with the general pattern of cluster 3. Thus, cluster 3 was labeled the *Curious Strategy*.

Finally, cluster 4 exemplar sequences showed that students in this group tended to conduct a large number of experiments, and they could produce the correct tables or graphs in their final attempt to solve the problem. They may have conducted more experiments after they produced the correct table or graph, but unlike cluster 3, they did not produce incorrect tables or graphs after the correct ones. Not surprisingly, these students had the highest probability of correctly answering the question (.72). As seen before, most of the sequential patterns were subsequences of the exemplar sequences, with the exception of [0,1], and [1, 0, 0]. Some example sequences include [0,1], and [1, 0, 0] are [0, 1, 0, 0], and [0, 4, 1, 0, 0, 0, 0, 0]. In these examples, students did not make wrong tables or graphs after they produced the correct graph. Consequently, cluster 4 was labeled as the *optimal* group.

Table 4. *Exemplar sequences and sequential patterns for each identified cluster*.

Cluster	Exemplar sequence	Sequential patterns
1	[0] [0] [0]	[0]
2	[0, 2, 3, 4, 0, 0, 0] [0, 2, 4, 2, 4, 4, 2, 0, 2, 0, 0, 0, 0]	[2, 0] [0, 4, 2]

	[0, 2, 4, 4, 4, 0, 0]	[0, 0, 0, 0] [0, 4, 0]
3	[0, 4, 2, 4, 0, 3, 3, 2, 2] [0, 0, 0, 4, 3, 2, 0] [0, 3, 3, 0, 0, 0, 0, 0, 0, 0, 0, 2]	[0, 4, 2] [3, 0] [0, 1] [0, 0, 0, 0, 2] [0, 4, 0] [0, 3, 2] [0, 0, 0, 0, 0]
4	[0, 0, 3, 0, 0, 0, 0] [3, 0, 0, 0, 0, 0] [0, 4, 2, 4, 0, 0, 0, 3, 0, 0, 0, 0, 0, 0]	[0, 0, 0, 0, 0, 0, 0, 0, 0] [0, 1] [0, 3, 0, 0, 0] [1, 0, 0] [3, 0, 0, 0, 0]

Note: 0: select a payload mass and run the experiment; 1: make a correct graph; 2: make an incorrect graph; 3: make a correct table; 4: make an incorrect table

The results of the analysis suggested the importance of order when clustering students' problem-solving sequences. If the frequency or occurrence of the actions was examined, cluster 2, 3 and 4 might appear to be similar to each other, since they all conducted several experiments, made the correct as well as incorrect graphs and tables. However, in cluster 2, students often made the correct tables or graphs initially, but **immediately** changed to generating incorrect tables/graphs. In cluster 3, students tended to produce the correct table/graph initially, and then they would conduct several experiments before they switched to generating the incorrect tables/graphs. In cluster 4, students seemed to make incorrect tables/graphs in their initial attempts, but then they produced the correct tables/graphs in the end. The order of the actions determined the probability of correctly answering the question.

RQ6. Did the LSTM cluster and review identify the same strategies? If not, what are the differences?

Similar to the review, LSTM cluster identified the same incorrect strategies (i.e., random try & guess strategy), but unlike the review, LSTM cluster grouped correct strategies in different

ways. Instead of clustering students based on their use of graphs and tables, LSTM cluster grouped students based on the order they used these tools. While this difference could be attributed to the instability of small sample size estimation, it could also be due to the fact that the review done by the author was based on intuitive understanding not statistics, and consequently, the clusters were formed based on surface features rather than statistically important features.

The results also have implications for the technical usage of the Gaussian mixture model and sequential pattern analysis. Although Gaussian mixture model can be used to help determine the number of clusters, convergence can be an issue. This is likely due to the fact that LSTM hidden unit values may not meet the assumptions of the Gaussian mixture model. Multicollinearity, small variance, and violation of multivariate normality could all cause convergence issues for Gaussian mixture model. These problems are likely to exist because we have little control over the properties of LSTM hidden units, which are part of the "black box" of neural networks. Despite the estimation difficulties of Gaussian mixture model, its results were consistent with the visual inspection of the LSTM output unit histogram and *k*-mean cluster analysis. Therefore, in practice, one recommendation would be the use of the LSTM output unit histogram to determine the number of clusters, and *k*-mean cluster analysis to determine the class membership of individuals.

The results also showed that for sequential pattern analysis, the resulting sequential patterns may not be interpretable if they are examined in isolation. However, they become more interpretable when they are examined together with exemplar sequences.

Chapter 5 A Simulation Study

In order to evaluate the accuracy of the proposed LTSM cluster under different conditions, a simulation study was conducted as it offers the flexibility of specifying and manipulating values of various parameters and examining their effects on the accuracy of the LTSM cluster. To mimic the real aspects of student problem solving, the simulation was designed based on the real log files discussed previously. Artificial log data were generated based on the five strategies identified from the review of the real log files (discussed in Chapter 4). I chose to simulate the five strategies identified from the review instead of the four strategies identified from the LSTM cluster analysis because the five strategies reflect more complexity, and consequently result in more difficult simulation tasks. However, this decision is arbitrary. The goal was to test the accuracy of LSTM cluster at recovering the true cluster membership of students using known strategies. More specifically, the research questions were:

7) How does sample size and number of actions influence the LSTM training, validating, testing, and clustering error rates?
8) What are the practical recommendations for using LSTM cluster based on the simulation results?

Method

In this simulation study, two factors were considered: sample size and problem complexity (i.e., the number of possible actions for solving the problem). The goal was to explore the sample size requirement for the LSTM cluster and how this is related to the problem complexity as reflected by the number of possible actions in solving the problem. I manipulated 3 levels of sample sizes (i.e., 100/strategy, 500/strategy & 1000/strategy) to reflect small, medium and large sample sizes. It is important to note that for each dataset, there were three sub-

datasets: training, validation, and testing sub-datasets. For each sub-dataset, five strategies were simulated. Each strategy had a sample size corresponding to its simulation condition. For example, in the 100/strategy condition, with a total of five strategies and three sub-data sets, the full dataset would contain 100*5*3=1500 samples. While in practice, the validation and testing datasets do not have to be as large as the training dataset, in this simulation study, the same sample size was chosen to ensure the accuracy of the estimation of validation and testing dataset error rates. The number of actions was set at 23 (the list of possible actions shown in Table 2) or 5 (the combined actions used in the above LSTM cluster analysis). In total, there were six simulation conditions. For each simulation condition, one hundred sets of log data were generated based on the five problem-solving strategies identified from the review of log files. The detailed data generation procedure for each type of problem-solving strategy is described in the following sections.

Graph strategy. Graph strategy was operationalized as problem-solving sequences that 1) contain 4 to 12 experiments with at least 3 unique experiments, and 2) contain the correct graph subsequence after any incorrect graph or table subsequences. To add more complexity to the dataset, the graph strategy may include the correct table subsequence at the beginning of the sequence, but it must be immediately followed by an incorrect table subsequence. This is because in the real data study, it was observed that if a correct table is immediately followed by an incorrect table at the beginning of the problem-solving sequence, then students cannot deduce the answer from the correct table. More specifically, in the 23 action conditions, each experiment was represented using the subsequence, [selectedMass_X, tryit] (i.e., select a specific pay load mass and conduct the experiment; for detailed meaning of these codes, see Table 2), where the X in selectedMass_X can take any value from the list, [10, 20, 30, 40, 50, 60, 70, 80, 90]. The

correct graph subsequence was [makeGraph, vertAxis_altitude, horizAxis_mass]. The incorrect graph subsequences included the action 'makeGraph' and any **combinations** of two elements from the list, ['vertAxis_altitude', 'horizAxis_mass', 'horizAxis_helium', and 'vertAxis_time'] that are not [vertAxis_altitude, horizAxis_mass]. The correct table subsequences included the 'make_table' action and any combination of 'selectedTableVars_Altitude_(ft.)', 'selectedTableVars_Payload_Mass_(lbs.)', 'selectedTableVars_Amount_of_Helium(cu.ft.)', and 'selectedTableVars_Time to Final_Altitude_(min.)' that are the supersets of [selectedTableVars_Altitude_(ft.), selectedTableVars_Payload_Mass_(lbs.)]. The incorrect table subsequences included the action 'Make_Table' and any combinations of 'selectedTableVars_Altitude_(ft.)', 'selectedTableVars_Payload_Mass_(lbs.)', 'selectedTableVars_Amount_of_Helium(cu.ft.)', and 'selectedTableVars_Time to Final_Altitude_(min.)' that are not the supersets of [selectedTableVars_Altitude_(ft.), selectedTableVars_Payload_Mass_(lbs.)]. All sequences must also include the 'Begin_problem1' action in the beginning, and 'Problem1Question2' action in the end.

The data generation algorithm can be briefly described as follows: 1) generate *n* experiment subsequences with at least 3 unique experiment subsequences, where n is an integer ranging from 4 to 12; 2) generate *x* incorrect graph subsequences, where *x* is an integer ranging from 0 to 2; 3) generate y incorrect table subsequences, where y is an integer ranging from 0 to 2; 4) randomly shuffle the generated subsequences; 5) insert the correct graph subsequence anywhere after the last incorrect graph or table subsequence; 6) randomly generate a binary number using Bernoulli distribution with a probability of 0.50, and if the number equals to one, insert the correct table in the first or second position in the sequence, followed by inserting an incorrect table subsequence; 7) insert the 'Begin_problem1' action in the beginning of the sequence and

the ‘Problem1Question2’ action in the end of the sequence. The multiple-choice item scores for the graphing strategy sequences were generated using a Bernoulli distribution with a probability of 0.73. In the 5-action conditions, the actions in each subsequence described above (i.e., experiment, correct graph, incorrect graph, correct table, & incorrect table) were combined to form a single action, and action ‘Begin_problem1’ and ‘Problem1Question2’ were removed because every student performed these actions. There were only five possible actions: ‘experiment’, ‘correct graph’, ‘incorrect graph’, ‘correct table’, and ‘incorrect table’. The same data generation algorithm was used to generate the data.

Table strategy. The table strategy was operationally defined as any sequences that include 4 to 12 experiments with at least 3 unique experiments and the correct table subsequence after any incorrect graph or table subsequences. Again, to add complexity to the simulation, the table strategy sequence could include the correct table subsequence at the beginning but it needed to be immediately followed by an incorrect table subsequence. This way, the first correct table was ‘erased’ by the second table, and it is equivalent to producing no correct table. The procedure to generate the table strategy was the same as the procedure to generate the graph strategy, except that correct graph was replaced by the correct table subsequence. The multiple-choice item scores for the table strategy sequences were generated using a Bernoulli distribution with a probability of 0.50.

Try-them-all strategy. The try-them-all strategy was operationally defined as any sequences that include 4 to 12 experiments with at least 3 unique experiments and both correct table and correct graph subsequences after any incorrect graph or table subsequences. The try-them-all strategy sequence could include the correct graph or table subsequences at the beginning of the sequence, but it needed to be immediately followed by the corresponding

incorrect graph or table subsequences. The data generation procedure was similar to the graph and table strategy data generation procedure described previously except that in step 5, both the correct table and correct graph subsequences were randomly inserted after the last incorrect table or graph subsequence. The multiple-choice item scores for the try-them-all strategy sequences were generated using a Bernoulli distribution with a probability of .96.

Random try strategy. The random try strategy was operationally defined as any sequences that included 4 to 12 experiments with at least three unique experiments, 1 to 2 wrong graphs or tables, and zero or one correct graph or table that occurred at the beginning of the sequence but was immediately followed by an incorrect graph or table. The data generation algorithm was similar to the previous algorithms except that no correct graph or table was generated near the end of the sequence. The multiple-choice item scores for the random try strategy sequences were generated using a Bernoulli distribution with a probability of 0.27.

Guess strategy. The guess strategy was operationally defined as any sequences that included 0 to 2 experiments, 1 to 2 wrong graphs, 0 to 2 wrong tables, and 0 to 1 correct graphs. The data was generated using the following algorithm: 1) generate 0 to 2 experiment subsequences, 2) generate 1 to 2 wrong graphs, 3) generate 0 to 2 wrong tables, 4) generate 0 to 1 correct graph, 5) randomly shuffle the generated sequence, 6) insert the 'Begin_problem1' action at the beginning of the sequence and the 'Problem1Question2' action at the end of the sequence. The multiple-choice item scores for the guess strategy sequences were generated using a Bernoulli distribution with a probability of .04.

Analysis. The generated data were analyzed using the first two steps of LSTM cluster (i.e., LSTM and cluster analysis). For 23-action conditions, 12 LSTM hidden units/cells were

used. For 5-action conditions, 10 LSTM hidden units/cells were used. These numbers were experimentally determined using trial datasets. The number of hidden units that led to the lowest error rate in the validation dataset was chosen. For example, in order to determine the number of LSTM hidden units for 23-action conditions, a trial dataset with sample size of 1000 per strategy was first simulated. Each trial dataset contained three sub-datasets: training, validation, and testing datasets.

Next, LSTM networks with 2 to 20 hidden units were trained, and their prediction error rates in validation dataset were recorded. Since the validation dataset error rates flattened after 12 hidden unit model, the number of hidden units was set to 12. The adadelta mini-batch gradient descent algorithm was used to optimize the LSTM. Dropout algorithm and weight decay regularization algorithms were used to prevent overfitting. The decay parameter for the weight decay regularization was set to 0.01.

After training the LSTM, *k*-mean cluster analysis was applied on the LSTM output unit probabilities to cluster the sequences. The number of clusters was set based on examples of LSTM output unit probability histograms. This is because it was not feasible to produce histograms for every simulation dataset. As shown later in the chapter, since most of the examples of LSTM output unit probability histograms suggested that there were 5 clusters, the default number of clusters was set to be 5. Also, *K*-mean cluster analysis was chosen over Gaussian mixture model because of its faster computational speed and that real data study revealed that the two procedures were comparable.

Evaluation criteria. To evaluate the accuracy of the proposed LSTM cluster method, the prediction error rates of the training, validating, and testing datasets were examined. Since in

data generation, each strategy was associated with a probability to correctly answer the multiple-choice question, the expected prediction error rate can be calculated. For the strategies that were more likely to lead to the correct answer to the final task (i.e., graph, table and try-them-all strategies), the probability for a correct response were set at .73, .50, and .96, respectively. Therefore, their expected prediction error rates were 27%, 50%, and 4% respectively. For the two strategies (i.e., random try strategy & guess strategy) that are more likely to result in the wrong answer to the final task, the probability for a correct response was set at .27 and .04 respectively. Therefore, their expected prediction error rates were 27% and 4% respectively. In total, the overall expected error rate was (27%+50%+4%+27%+4%)/5=22.4% given that the five strategies were manipulated to be equally distributed. This means ideally, LSTM's prediction error rates in the training, validating, and testing datasets should be close to 22.4%. In addition to LSTM prediction error rates, the clustering error rates for the entire dataset (i.e., training + validation + testing sub-datasets) were also examined. Since *k*-mean cluster analysis algorithm only clusters simulated problem-solving sequences into groups without assigning names to clusters, each cluster need to be named according to the five strategies initially used in the simulation design. This is required to determine the number of cases that were correctly clustered.

To name the clusters resulting from the cluster analysis, a two-step algorithm was used. First, a 5-by-5 frequency matrix was calculated for each simulated dataset. The 5 rows of the frequency matrix represented the memberships of 5 expected clusters. The 5 columns of the frequency matrix represented the memberships of the five 5 observed clusters. The elements in the matrix were frequency counts. For example, the element in the first row and second column of the matrix represented the number of simulated cluster 1 sequences that were clustered as belonging to observed cluster 2. It is important to note that at this stage, it was unclear which

observed cluster corresponded to which expected cluster. In the next step, five elements with unique row and column numbers from the frequency matrix were selected so that their sum was larger than any other sets of such elements. Since there were only 5!=120 possible combinations, we simply calculated the sums for each of the 120 possible combinations, and identified the combination with the largest sum. This sum represents the number of cases that were correctly clustered. Using this max sum, the cluster error rate was defined by the following formula:

$$cluster\ error\ rate = 1 - \frac{\max sum}{total\ sample\ size}. \quad (23)$$

Results

RQ1. How did sample size and number of actions influence the LSTM training, validating, testing, and clustering error rates?

The means and standard deviations of LSTM prediction error rates and clustering error rates were reported in Table 5. As can be seen, the prediction error rates for the training, validating, and testing datasets were close to the expected error rate 22.4% for the 500 and 1000/strategy sample size conditions. However, in 100 sample size 22-action condition, the prediction error rate for the training dataset was lower than 22.4% (16.8%), and the prediction error rates for validation and testing datasets were higher than 22.4% (24.6% and 26.1% respectively). This indicated that with small sample size, LSTM is vulnerable to overfitting. The problem of overfitting was reduced in the 5-action condition: the prediction error rate for the training, validating, and testing datasets were 19.8%, 22.4% and 23.4%, respectively. Although overfitting did occur, the degree was smaller than that in the 100 sample size and, 22-action condition.

Examples of LSTM output unit probability histograms for each simulation condition are shown in Figure 11. As can be seen, most histograms showed five major peaks corresponding to the five clusters of strategies. The trend was that as sample size increased and action number decreased, the peaks became sharper. It is important to note that under the conditions with the sample size of 100 and 23 actions, there were only two major peaks, indicating that the LSTM network did not capture the differences among the five clusters. Since the majority of the histograms showed five major peaks, in *k*-mean cluster analysis, the number of means was set to five.

Consistent with the trends seen in LSTM prediction error rates and LSTM output unit probability histograms, the mean clustering error rates decreased as the sample size increased and action number decreased. The highest mean clustering error rate was seen in the 100 sample size and 23 action condition (45.3%), and the lowest mean cluster error rate was observed in the 1000 sample size 23 action condition (11.2%). The standard error of the mean tended to increase as sample size and number of action decreased.

RQ2. What are the practical recommendations for using LSTM cluster based on the simulation results?

Overall, the simulation suggested that LSTM cluster worked well with large sample sizes (n=1000 per group). With small sample sizes, it is recommended to reduce the complexity of the problem by eliminating nonessential actions, or combining actions closely related to each other.

Table 5. *Mean prediction and cluster error rates for all simulation conditions.*

Complexity	Sample size	Train error	Valid error	Test error	Cluster error
5 actions	1000	22.2 (.009)	22.4 (.005)	22.8 (.006)	12.1 (.056)
5 actions	500	22.0 (.010)	22.4 (.008)	22.9 (.008)	16.3 (.062)
5 actions	100	19.8 (.037)	22.4 (.016)	23.4 (.019)	31.8 (.090)
22 actions	1000	21.3 (.007)	22.4 (.005)	22.8 (.006)	11.2 (.030)
22 actions	500	21.0 (.011)	22.5 (.007)	23.2 (.008)	20.9 (.058)
22 actions	100	16.8 (.043)	24.6 (.018)	26.1 (.022)	45.3 (.061)

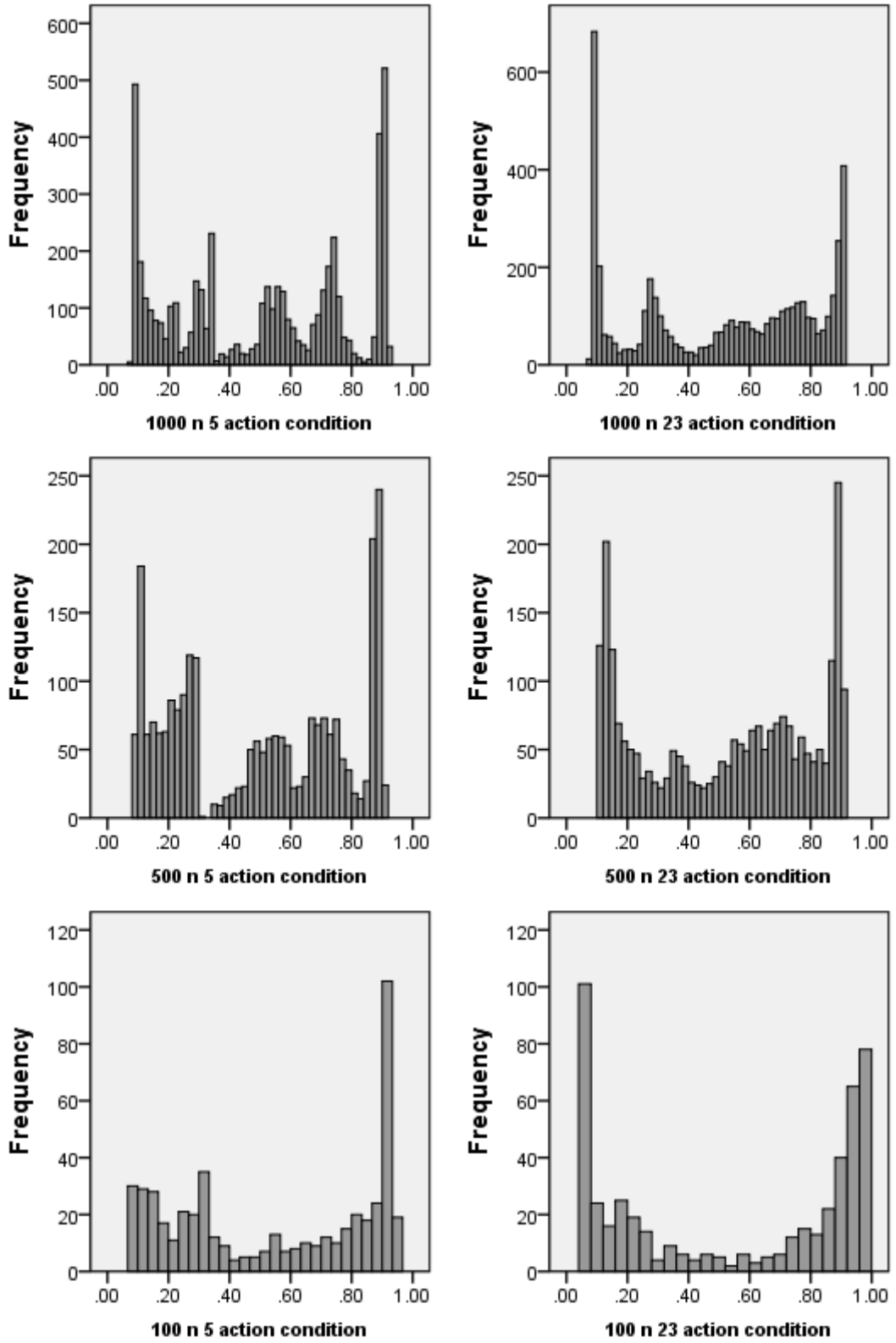

Figure 11. Examples of LSTM output unit probability histograms for all six simulation conditions.

Chapter 6 Discussion

Implications

Only very recently have researchers begun to apply deep learning neural networks in the field of educational data mining (Huang & Brusilovsky, 2016; Piech et al., 2015). This study is an attempt to bring deep learning technology into educational data mining for the analysis of log data. It provides a way to integrate several current educational data mining methods in the analysis of log files to facilitate the classification of students' problem-solving strategies. The results of the simulation and real data studies suggested that the proposed method, LSTM cluster, is a viable approach in clustering students' problem-solving sequences.

In order for the LSTM cluster to be effective, however, researchers need to evaluate whether the sample size is sufficient relative to the complexity of the problem. Ideally, like any other deep learning neural networks, LSTM cluster should be used when large samples are available. When only a small sample size is available (e.g., a few hundreds), content experts should reduce the complexity of the log data by removing theoretically irrelevant actions and combining theoretically closely related actions prior to the use of LSTM cluster. Researchers can roughly estimate the "fit" of LSTM cluster by comparing training dataset prediction error rates with testing dataset prediction error rates. If the error rates were similar between the training and testing datasets, and the errors are reasonably low, then LSTM can be applied. If there is a large discrepancy (e.g., >5%) between the training and testing prediction error rates, then researchers can choose to collect more samples, or reduce the complexity of the log files. If the prediction error rates are high in general, then researchers can increase the number of hidden nodes to improve the prediction accuracy.

This study demonstrated that a classification tool such as LSTM (or other neural networks) not only is useful for predicting the outcome variables, but also can be indirectly used to solve clustering problems. This can be achieved by clustering the hidden units (extracted features) of the neural network. To understand the differences among clusters, the exemplar input for each cluster could be identified and examined, which can serve as a simple way to understand the "black box" of the artificial neural networks and other machine learning techniques.

In addition, the findings of this study suggested that order of student problem-solving steps is indeed important for understanding the problem-solving strategies. Traditionally, when analyzing log data, only action frequencies are concerned. While this approach may work in some situations, analyses that ignore order could potentially produce misleading results. Action order matters because the meaning of an action could depend on what happen before and after the action. For example, in this study, a graph with the correct IV and DV is only helpful when the student conducts enough experiments before or after creating the graph. If a correct graph is immediately replaced by a wrong graph, then it has little impact on the problem-solving outcome. However, unlike solving a math problem, the order to solve a science simulation problem is much less restricted. This makes heuristic-based sequence analysis methods such as sequential pattern mining less effective because they reply on fixed orders of actions.

Limitations and Recommendations for Future Research

There were three limitations to this study. First, at the time the study was conducted, there was no other sequence cluster methods for log files. However, at the end of the study, several comparable sequence cluster methods were proposed at the 10th International Conference on Educational Data Mining, including dynamic time warping (Shen & Chi, 2017) and hidden

Markov Models (Hansen et al., 2017). Future studies are needed to compare different sequence cluster methods and evaluate the strengths and weaknesses of these methods.

Second, the small sample size for the real data study weakened the validity of the results. As suggested by the simulation study, LSTM tended to be less generalizable (as indicated by low training error rates & high testing error rates) with small sample size conditions. Although this problem can be alleviated to a certain degree by reducing the complexity of the data (e.g., number of actions), the small sample size could still lead to larger cluster error rates. Consequently, the real data study needs to be interpreted with caution. In future research, more participants are needed to test the validity of the results.

Third, LSTM cluster may need even larger sample sizes if there are a large number of clusters of student strategies. This is because with a large number of clusters, the cluster means are more likely to be close to each other. In order to accurately separate two clusters, a larger sample size may be required. Alternatively, different cluster analysis methods such as hierarchical cluster may be used for this situation. In this simulation study, "number of clusters" was not manipulated. Consequently, it was unclear how the number of clusters may impact the sample size requirement. In future simulation study, the effect of number of clusters on accuracy of LSTM cluster can be tested.

Significance

This study proposed a comprehensive method to cluster student log files, which is important for several reasons. First, it helps researchers and instructors better understand students' problem strategies/processes. Second, it provides a data driven approach to identify important indicators and latent constructs, which are essential for any measurement model. This data driven

approach can be combined with theory driven approach to develop better measurement model. Third, clustering students is essential for personalizing the digital learning environment for each student. More specifically, clustering students' problem solving strategies can help the computer education program to provide more relevant feedbacks, and make more meaningful recommendations. Last, the clustering results can be used to evaluate the validity of the assessment (i.e., does the assessment indeed elicit behaviors the assessment intends to elicit from students). This information can be used to improve the assessment.

REFERENCES

Agrawal, R., & Srikant, R. (1995). Mining sequential patterns: Generalizations and performance improvements. In P. S. Yu, & A. S. P. Chen (Eds.), *Proceedings of the 5th International Conference on Extending Database Technology* (pp. 3-14). Washington: IEEE Comput. Soc. Press.

Akaike, H. (1971). Information theory and an extension of the maximum likelihood principle. *Second International Symposium on Information Theory (Tsahkadsor, 1971).* Retrieved from http://login.ezproxy.library.ualberta.ca/login?url=http://search.ebscohost.com/login.aspx?direct=true&db=msn&AN=MR0483125&site=eds-live&scope=site; http://www.ams.org/mathscinet/MRAuthorID/215376; http://www.ams.org/mathscinet-getitem?mr=0483125

Amershi, S., & Conati, C. (2011). Automatic recognition of learner types in exploratory learning environments. In C. Romero, S. Ventura, M. Pechenizkiy, & R. S. J. d. Baker (Eds.), *Handbook of Educational Data Mining* (pp. 93-106). Boca Raton, FL: CRC Press.

Baker, E. L., Chung, G. K. W. K., & Delacruz, G. C. (2011). The best and future uses of assessment in games. In M. C. Mayrath, J. Clarke-Midura, D. H. Robinson, & G. Schraw (Eds.), *Technology-based assessments for 21st century skills: theoretical and practical implications from modern research* (pp. 229-248). Charlotte, NC: Information Age Publishing, Inc.

Barker, K., Trafalis, T., & Rhoads, T. R. (2004). *Learning from student data*. Paper presented at Proceedings of the 2004 IEEE Systems and Information Engineering Design Symposium. Charlottesville, VA: University of Virginia.

Bennett, R. E., Persky, H., Weiss, A. R. & Jenkins, F. (2007). *Problem solving in technology-rich environments: a report from the NAEP technology-based assessment project* (NCES 2007-466). U.S. Department of education, DC: National Center for Education Statistics. Retrieved from the institute of education science website: http://nces.ed.gov/nationsreportcard/pubs/studies/2007466.asp

Behrens, J. T., Mislevy, R. J., DiCerbo, K. R., & Levy, R. (2011). Evidence centered design for learning and assessment in the digital world. In M. C. Mayrath, J. Clarke-Midura, D. H. Robinson, & G. Schraw (Eds.), *Technology-based assessments for 21st century skills: theoretical and practical implications from modern research* (pp. 13-54). Charlotte, NC: Information Age Publishing, Inc.

Bishop, C. M., Svensén, M., & Williams, C. K. I. (1998) GTM: the generative topographic mapping. *Neural Computing, 10*(1), 215-234.

Buckley, B. C., Gobert, J. D., & Horwitz, P. (1999). Using log files to track students' model-based inquiry. *Journal of Management, 25*(1), 1-27.

Capterra. (2015, Dec). *Top LMS Software*. Retrieved from: http://www.capterra.com/learning-management-system-software/#infographic

Carrier, P. L., & Cho, K. (2016, Sep 9). LSTM networks for sentiment analysis. Retrieved from http://deeplearning.net/tutorial/lstm.html

Ciresan, D. C., Meier, J., & Schmidhuber, J. (2012). *Multi-column deep neural networks for image classification*. Paper presented at IEEE conference on computer vision and pattern recognition (CVPR2012), Rhode Island.

Christodoulopoulos, C. E., & Papanikolaou, K. A. (2007). A group formation tool in a e-learning context. In *The 19th IEEE International Conference on Tools with Artificial Intelligence*. Patras, Greece: IEEE Computer Society.

Chu, M. (2016). Using computer simulated science laboratories: a test of pre-laboratory activities with the learning error and formative feedback model (Doctoral dissertation). Retrieved from *Electronic Theses and Dissertations (University of Alberta)*. doi: https://doi.org/10.7939/R3416TB42

Clarke-Midura, J., Code, J., Dede, C., Mayrath, M., & Zap, N. (2011). Thinking outside the bubble: virtual performance assessments for measuring complex learning. In M. C. Mayrath, J. Clarke-Midura, D. H. Robinson, & G. Schraw (Eds.), *Technology-based assessments for 21st century skills: theoretical and practical implications from modern research* (pp. 13-54). Charlotte, NC: Information Age Publishing, Inc.

Cocea, M., & Weibelzahl, S. (2006). *Can log files analysis estimate learners' level of motivation?* Paper presented at Proceedings of Lernen-Wissensentdeckung-Adaptivität (LWA2006), Hildesheim, Germany.

Dagenais, B. (2015). Pymining - a collection of data mining algorithms in Python (Version 0.2). GitHub repository, https://github.com/bartdag/pymining.

De Klerk, S., Veldkamp, B. P., & Eggen, T. J. H. M. (2015). Psychometric analysis of the performance ddata of simulation-based assessment: a systematic review and a Bayesian network example. *Computers & Education, 85*, 23-34.

Desmarais, M. C., & Pu, X. (2005). A Bayesian student model without hidden nodes and its comparison with item response theory. *International Journal of Artificial Intelligence in Education, 15*， 291-323.

Deng, L., Hinton, G., & Kingsbury, B. (2013). *New types of deep neural network learning for speech recognition and related applications: an overview*. Paper presented at IEEE International Conference on Acoustic Speech and Signal Processing (ICASSP 2013), Vancouver, BC.

Deng, L., Yu, D. (2014). Deep learning: methods and applications. *Foundations and Trends in Signal Processing, 7*(3-4): 1-199. doi:10.1561/2000000039

Dunn, J. C. (1973). A fuzzy relative of the ISODATA process and its use in detecting compact well-separated clusters. *Cybernetics and Systems, 3,* 32-57. doi: 10.1080/01969727308546046

Ellis, R. K. (2009). *Field Guide to Learning Mangagement Systems*. ASTD Learning Circuits.

Escueta, M., Quan, V., Nickow, A. J., & Oreopoulos, P. (2017). *Education technology: an evidence-based review* (No. w23744). National Bureau of Economic Research.

Etheredge, M., Lopes, R., & Bidarra, R. (2013). A generic method for classification of player behavior. In M. J. Nelson, A. M. Smith, & G. Smith (Eds.), *Proceedings of the Second*

AIIDE Workshop on Artificial Intelligence in the Game Design Process (pp. 56-61). New York, NY: CM Press.

Everitt, B. (2011). *Cluster analysis* (5th ed.). Chichester, West Sussex, U.K.: Wiley. Retrieved from http://proquest.safaribooksonline.com/?uiCode=ualberta&xmlId=9780470978443

Felder, R. M. and Soloman, B. A. (). *Index of learning style questionnaire (ILSQ).* Retrieved from http://www.ncsu.edu/felder-public/ILSdir/styles.htm

Fukushima, K. (1980). Neocognitron: A self-organizing neural network model for mechanism of pattern recognition unaffected by shift in position. *Biol. Cybern. 36*, 193-202. doi: 10.1007/bf00344251

García, E., Romero, C., Ventura, S., de Castro, C., & Calders, T. (2011). Association rule mining in learning management systems. In C. Romero, S. Ventura, M. Pechenizkiy, & R. S. J. d. Baker (Eds.), *Handbook of Educational Data Mining* (pp. 93-106). Boca Raton, FL: CRC Press.

Gee. J. P. (2008). Video games and embodiment. *Games and Culture, 3*(3-4), 253-263. doi: 10.1177/1555412008317309

Geiser, C. (2010). *Data Analysis with Mplus*. New York: The Guildford Press.

Gers, F., Schraudolph, N., Schmidhuber, J. (2002). Learning precise timing with LSTM recurrent networks. *Journal of Machine Learning Research, 3*, 115-143.

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning.* MIT Press.

Goutte, C., Hansen, L. K., Liptrot, M. G., & Rostrup, E. (2001). Feature-space clustering for fMRI meta-analysis. *Human Brain Mapping, 13*(3), 165-183. doi:10.1002/hbm.1031

Hämäläinen, W., & Vinni, M. (2011). Classifiers for educational data mining. In C. Romero, S. Ventura, M. Pechenizkiy, & R. S. J. d. Baker (Eds.), *Handbook of Educational Data Mining* (pp. 93-106). Boca Raton, FL: CRC Press.

Hansen, C., Hansen, C., Hjuler, N., Alstrup, S., & Lioma, C. (2017, June). Sequence modelling for analysing student interaction with educational systems. Paper presented at the 10th international conference on educational data mining. http://educationaldatamining.org/EDM2017/proceedings-full/

Harpstead, E., MacLellan, C. J., Koedinger, K. R., Aleven, V., Dow, S. P., & Myers, B. A. (2013, July). Investigating the solution space of an open-ended educational game using conceptual feature extraction. In S. K. D'Mello, R. A. Calvo, & A. Olney (Eds), *Proceedings of the 6th International Conference on Educational Data Mining* (pp. 51-58). Memphis, TN: International Educational Data Mining Society.

Hershkovitz, A. & Nachmias, R. (2011). Log-based assessment of motivation in online learning. In C. Romero, S. Ventura, M. Pechenizkiy, & R. S. J. d. Baker (Eds.), *Handbook of Educational Data Mining* (pp. 93-106). Boca Raton, FL: CRC Press.

Hinton, G. E., Dayan, P., Frey, B. J., & Neal, r. (1995). The wake-sleep algorithm for unsupervised neural networks. *Science, 268* (5214): 1158-1161. doi: 10.1126/science.7761831

Hinton, G., Deng, L., Yu, D., Dahl, G., Mohamed, A., Jaitly, N., Senior, A., Vanhoucke, V., Nguyen, P., Sainath, T., Kingsbury, B. (2012). Deep neural networks for acoustic modeling in speech recognition – the shared views of four research groups. *IEEE Signal Prcessing Magazine, 29*(6), 82-97.

Hinton, G., Salakhutdinov, R. (2006). Reducing the dimensionality of data with neural networks. *Science, 313,* 504-507. doi: 10.1126/science.1127647

Hinton, G. E. Srivastava, N., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. R. (2012). Improving neural networks by preventing co-adaptation of feature detectors. Retrieved from: https://arxiv.org/pdf/1207.0580.pdf

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation, 9*(8), 1735-1780.

Huang, Y., & Brusilovsky, P. (2016). Towards modeling chunks in a knowledge tracing framework for students' deep learning. In T. Barnes, M. Chi, & M. Feng (Eds.), *Proceedings of the 9th International Conference on Educational Data Mining* (pp. 666-668), Raleigh, North Carolina.

Ioannidou, A., Repenning, A., Webb, D., Keyser, D., Luhn, L. & Daetwyler, C. (2010). Mr. Vetro: A collective simulation for teaching health science. *Computer-Supported Collaborative Learning, 5,* 141-166. doi: 10.1007/s11412-010-9082-8

Ivakhnenko, A. (1965). *Cybernetic Predicting Devices.* Kiev: Naukova Dumka.

Ivakhnenko, A. (1971). Polynomial theory of complex systems. *IEEE Transactions on Systems, Man and Cybernetics* 4: 364-378.

Kay, J., Maisonneuve, N., Yacef, K., & Zaiane, O. R. (2006). Mining patterns of events in students' teamwork data. In *Proceedings of Educational Data Mining Workshop* (pp. 1-8), Taiwan.

Kerr, D., & Chung, G. K. W. K. (2012). Identifying key features of student performance in educational video games and simulations through cluster analysis. *Journal of Educational Data Mining, 4* (1), 144-182.

Kerr, D. (2015). Using data mining results to improve educational video game design. *Journal of Educational Data Mining, 7*(3), 2015.

Khajah, M., Lindsey, R. & Mozer, M. (2016). How deep is knowledge tracing? In T. Barnes, M. Chi, & M. Feng (Eds.), *Proceedings of the 9th International Conference on Educational Data Mining* (pp. 666-668), Raleigh, North Carolina.

Khalil, F., Li, J., & Wang, H. (2008). Integrating recommendation models for improved web page prediction accuracy. In The 31st Australasian Computer Science Conference (pp. 91-100), Wollongong, Australia.

Koedinger, K. & Corbett, A. (2006). Cognitive tutors: Technology bringing learning science to the classroom. In K. Sawyer (Ed.), *The Cambridge Handbook of the Learning Sciences.* Cambridge, U.K.: Cambridge University Press, pp. 61-78.

Kohonen, T. *Self-Organizing Maps*, 3rd edn. Berlin, Germany: Springer-Verlag.

Lance, G.N., & Williams, W. T. (1967). A general theory of classificatory sorting strategies: Hierarchical systems. *Computer Journal, 9,* 373-380.

Langeheine, R., Pannekoek, J., & van de Pol, F. (1996). Bootstrapping goodness-of-fit measures in categorical data analysis. *Sociological Methods and Research, 24,* 249-264.

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature, 521*, 436-444. doi: 10.1038/nature14539

Leighton, J. P., & Chu, M. (2016). First among equals: Hybridization of cognitive diagnostic assessment and evidence-centered game design. *International Journal of Testing, 16*(2), 164-180. Retrieved from http://login.ezproxy.library.ualberta.ca/login?url=http://search.ebscohost.com/login.aspx?direct=true&db=eric&AN=EJ1095667&site=eds-live&scope=site; http://dx.doi.org/10.1080/15305058.2015.1107075

Levy, R. (2012). *Psychometric advances, opportunities, and challenges for simulation-based assessment* (Center for K-12 Assessment and Performance Management Report). Educational Testing Service. Retrieved from http://www.ets.org/Media/Research/pdf/session2-levy-paper-tea2012.pdf

Levy, R. (2014). *Dynamic Bayesian network modeling of game based diagnostic assessments (CRESST report 837)*. Los Angeles, CA: University of California, National Center for Research on Evaluation, Standards, and Student Testing (CRESST).

Lo, Y., Mendell, N. R., & Rubin, D. B. (2001). Testing the number of components in a normal mixture. *Biometrika, 88*, 767-778.

Lu, F., Li, X., Liu,Q., Yang, Z., Tan, G., & He, T. (2007). Research on personalized e-learning system using fuzzy set based clustering algorithm. In Shi et al. (Eds.), *ICCS'2007* (587-590). Berlin, Germany: Springer-Verlag.

Luan, J. (2002). Data mining and its applications in higher education. *New Directions Institut. Res., 113*, 17-36.

Manikandan, C., Sundaram, M. A. S., & Mahesh, B. M. (2006, Dec). Collaborative e-learning for remote education an approach for realizing pervasive learning environments. In *2nd International Conference on Information and Automation*, Colombo, Srilanka (pp. 274-278).

Mayrath, M. C., Clarke-Midura, J., & Robinson, D. H., (2011). Introduction to technology-based assessments for 21st century skills. In M. C. Mayrath, J. Clarke-Midura, D. H. Robinson, & G. Schraw (Eds.), *Technology-based assessments for 21st century skills: theoretical and practical implications from modern research* (pp. 1-12). Charlotte, NC: Information Age Publishing, Inc.

Mayrath, M. C., Clarke-Midura, J., Robinson, D. H., & Schraw, G. (2011). *Technology-based assessments for 21st century skills: theoretical and practical implications from modern research.* Charlotte, NC: Information Age Publishing.

McLachlan, G. J. (2000). *Finite Mixture Models*. New York, NY: Wiley.

McLachlan, G. J. (1988). *Mixture Models: inference and applications to clustering*. New York, NY: Dekker.

Min, W., Wiggins, J. B., Pezzullo, L. G., Vail, A. K., Boyer, K. E., Mott, B. W., Frankosky, M. H., Wiebe, E. N., Lester, J. C., (2016). Predicting dialogue acts for intelligent virtual agents with multimodal student interaction data. In T. Barnes, M. Chi, & M. Feng (Eds.), *Proceedings of the 9th International Conference on Educational Data Mining* (pp. 215-222), Raleigh, North Carolina.

Minaei-Bidgoli, B., Kashy, D. A., Kortemeyer, G., & Punch, W. (2003). *Predicting student performance: An application of data mining methods with an educational web-based system*. Paper presented at Proceedings of 33rd Frontiers in Education Conference, Westminster, CO.

Mislevy, R. J. (2006). Cognitive psychology and educational assessment. In R. L. Brennan (Ed.), *Educational measurement* (4th ed., pp.257-305). Westport, CT: American Council on Education/Praeger Publishers.

Mislevy, R. J., Steinberg, L. S., Almond, R. G., Breyer, F. J., & Johnson, L. (2001). *Making sense of data from complex assessment (CRESST Report 538)*. Los Angeles, CA: University of California, National Center for Research on Evaluation, Standards, and Student Testing.

Mullier, D. (2003). A tutorial supervisor for automatic assessment in educational systems. *International Journal of e-Learning, 2*(1), 37-49.

Mullier, D., Moore, D., & Hobbs, D. (2001). A neural-network system for automatically assessing students. In P. Kommers, & G. Richards (Eds.), *World Conference on Educational Multimedia* (pp. 1366-1371). Chesapeake, VA: AACE.

Muthén, L. K. & Muthén, B. O. (1998-2015). Mplus User's Guide. Seventh Edition. Los Angeles, CA: Muthén & Muthén.

Mylonas,P., Tzouveili, P., & Kollias, S. (2007). Intelligent content adaptation in the framework of an integrated e-learning system. In *The International Workshop in Combing Intelligent and Adaptive Hypermedia Methods/Techniques in Web Based Education Systems* (pp. 59-66), Salzburg, Austria.

National Assessment of Educational Progress, (2009). *Complete task library*. Retrieved from http://www.nationsreportcard.gov/science_2009/ict_indepth_gr4.aspx

Nilakant, K., & Mitovic, A. (2005). Application of data mining in constraint-based intelligent tutoring systems. In C. K. Looi, G. I. McCalla, B. Bredeweg, & J. Breuker (Eds.), *Proceedings of the 2005 Conference on Artificial Intelligence in Education: Supporting Learning through Intelligent and Socially Informed Technology* (pp. 896-898). Amsterdam, Netherlands: IOS Press.

Nylund, K. L., Asparouhov, T., & Muthen, B. O. (2007). Deciding on the number of classes in latent class analysis and growth mixture modeling: A monte carlo simulation study. *Structural Equation Modeling: A Multidisciplinary Journal, 14*(4), 535-569. Retrieved from http://login.ezproxy.library.ualberta.ca/login?url=http://search.ebscohost.com/login.aspx?direct=true&db=eric&AN=EJ780617&site=eds-live&scope=site; http://www.informaworld.com/openurl?genre=article&id=doi:10.1080/10705510701575396

Pahl, C., & Donnellan, C. (2003). Data mining technology for the evaluation of web-based teaching and learning systems. In *Proceedings of Congress E-learning* (pp. 1-7), Montreal, Canada.

Pedregosa, F., Varoquaux, G. Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research, 12,* 2825-2830.

Pei, J., Han, J., Mortazavi-Asl, B., Pinto, H., Chen, Q., Dayal, U., & Hsu, M. C. (2001). PrefixSpan: Mining sequential patterns efficiently by prefix-projected pattern growth. In *Proceedings of the 2001 International Conference on Data Engineering* (pp. 215-224), Heidelberg, Germany.

Pelleg, D. & Moore, A. (1999). Accelerating exact *k*-means algorithms with geometric reasoning. In *the fifth international conference on knowledge discovery in databases* (pp. 277-281). Menlo Park, CA: AAAI Press.

Pham, D. T., Dimov, S. S., & Nguyen, C. D. (2005). Selection of K in K-means clustering. *Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science, 219*(1), 103-119. doi:10.1243/095440605X8298

Piech, C., Bassen, J., Huang, J., Ganguli, S., Sahami, M., Guibas, L., & Sohl-Dickstein, J. (2015). Deep knowledge tracing. In NIPS proceedings. Retrieved from: https://papers.nips.cc/paper/5654-deep-knowledge-tracing.pdf

Psotka, J., Mutter, S. A. (1988). *Intelligent Tutoring Systems: Lessons Learned.* Lawrence Erlbaum Associates.

Quellmalz, E. S., Davenport, J. L., Timms, M. J., Deboer, G. E., Jordan, K. A., Huang, C., & Buckley, B. C. (2013). Next-generation environments for assessing and promoting complex science learning. *Journal of Educational Psychology, 105*(4), 1100-1114.

Quellmalz, E. S., Timms, M. J., & Buckley, B. C., Davenport, J., Loveland, M., & Silberglitt, M. D. (2011). 21st Century dynamic assessment. In M. C. Mayrath, J. Clarke-Midura, D. H. Robinson, & G. Schraw (Eds.), *Technology-based assessments for 21st century skills: theoretical and practical implications from modern research* (pp. 55-90). Charlotte, NC: Information Age Publishing, Inc.

Quellmalz, E. S., Timms, M. J., Silberglitt, M. D., & Buckley, B. C. (2012). Science assessments for all: integrating science simulations into balanced state science assessment systems. *Journal of Research in Science Teaching, 49*(3), 363-393.

Rodrigo, M. M. T., Anglo, E. A., Sugay, J. O., & Baker, R. S. J. D. (2008). Use of unsupervised clustering to characterize learner behaviors and affective states while using an intelligent tutoring system. In T. W. Chan, G. Biswas, F. C. Chen, S. Chen, C. Chou, M. Jacobson, Kinshuk, F. Klett, C. K. Looi, T. Mitrovic, R. Mizoguchi, K. Nakabayashi, P. Reimann, S. Suthers, S. Yang, & J. C. Yang (Eds.), *Proceedings of the 16th International Conference on Computers in Education* (pp. 49-56). Taipei, Taiwan: Asia-Pacific Society for Computers in Education.

Romero, C., González, P., Ventura, S., del Jesus, M. J., & Herrera, F. (2009). Evolutionary algorithms for subgroup discovery in e-learning: a practical application using Moodle data. *Expert Systems with Applications, 36*, 1632-1644. doi: 10.1016/j.eswa.2007.11.026

Romero, C., Ventura, S., Zafra, A., de Bra, P. (2009). Applying web usage mining for personalizing hyperlinks in web-based adaptive educational systems. *Comput. Educ., 53*, 828-840.

Romero, C., Ventura, S., Pechenizkiy, M., & Baker, R. S. J. d. (2011). *Handbook of Educational Data Mining*. Boca Raton, FL: CRC Press.

Rosch, E. (1975). Cognitive reference points. *Cognitive Psychology, 7,* 532-547.

Schmidhuber, J. (1992). Learning complex, extended sequences using the principle of history compression. *Neural Computation, 4,* 234-242.

Schwarz, G. A. (1978). Estimating the dimension of a model. *The Annals of Statistics,* (2), 461. Retrieved from http://login.ezproxy.library.ualberta.ca/login?url=http://search.ebscohost.com/login.aspx?direct=true&db=edsjsr&AN=edsjsr.2958889&site=eds-live&scope=site

SciKits – main. (n.d.). Retrieved October 03, 2016 from http://scikits.appspot.com/

Sharma, A., Biswas, A., Gandhi, A., Patil, S., & Deshmukh, O. (2016). Livelinet: A multimodal deep recurrent neural network to predict liveliness in educational videos. In T. Barnes, M. Chi, & M. Feng (Eds.), *Proceedings of the 9th International Conference on Educational Data Mining* (pp. 215-222), Raleigh, North Carolina.

Sison, R., Numao, M., & Shimura, M. (2000). Multistrategy discovery and detection of novice programmer errors. *Machine Learning, 38,* 157-180.

Shaffer, D. W., & Gee, J. P. (2011). The right kind of GATE: computer games and the future of assessment. In M. C. Mayrath, J. Clarke-Midura, D. H. Robinson, & G. Schraw (Eds.), *Technology-based assessments for 21st century skills: theoretical and practical implications from modern research* (pp. 211-228). Charlotte, NC: Information Age Publishing, Inc.

Shen, S., & Chi, M. (2017, June). Clustering student sequential trajectories using dynamic time warping. Paper presented at the 10th international conference on educational data mining. http://educationaldatamining.org/EDM2017/proceedings-full/

Shute, V. J., Hansen, E. G., & Almond, R. G. (2008). *An assessment for learning system called ACED: Designing forlearning effectiveness and accessibility*. RR-07-26 (pp. 1-54), Princeton, NJ: ETS Research Report.

Shute, V. J., & Ventura, M. (2013). *Measuring and supporting learning in games: Stealth assessment*. Ambridge, MA: The MIT Press.

Smith, E., & Medin, D. (1999). The exemplar view. *Concepts: Core readings,* 207-209.

Slotta, J. D., & Chi, M. T. H. (2006). Helping students understand challenging topics in science through ontology training. *Cognition and Instruction, 24*(2), 261-289.

Steinhaus, H. (1957). “Sur la division des corps matériels en parties”. *Bulletin of the Polish Academy of Sciences* (in French), *4*(12), 801-804.

Talavera, L. & Gaudioso, E. (2004). Mining student data to characterize similar behavior groups in unstructured collaboration spaces. In *Workshop in Artificial Intelligence in Computer Supported Collaborative Learning* (pp. 17-22), Valencia, Spain.

Tang, T. Y., & Chan, K. C. (2002). Feature construction for student group forming based on their browsing behaviors in an e-learning system. In M. Ishizuka, & A. Sattar (Eds.), PRICAI'2002 (pp. 512-521). Berlin, Germany: Springer-Verlag.

Teng, C., Lin, C., Cheng, S., & Heh, J. (2004). Analyzing user behavior distribution on e-learning platform with techniques of clustering. In *Society for Information Technology and Teacher Education International Conference* (pp. 3052-3058), Atlanta, GA.

Tibshirani, R., Walther, G., & Hastie, T. (2001). Estimating the number of clusters in a data set via the gap statistic. *Journal of the Royal Statistical Society.Series B (Statistical Methodology),* (2), 411. Retrieved from http://login.ezproxy.library.ualberta.ca/login?url=http://search.ebscohost.com/login.aspx?direct=true&db=edsjsr&AN=edsjsr.2680607&site=eds-live&scope=site

Tian, F., Wang, S., Zheng, C., & Zheng, Q. (2008, Apr). Research on e-learner personality grouping based on fuzzy clustering analysis. In *The 12th International Conference on Computer Supported Cooperative Work in Design* (pp. 1035-1040), Xi'an, China.

Vellido, A., Castro, F., & Nebot, À. (2011). Clustering Educational Data. In C. Romero, S. Ventura, M. Pechenizkiy, & R. S. J. d. Baker (Eds.), *Handbook of Educational Data Mining* (pp. 93-106). Boca Raton, FL: CRC Press.

Vellido, A., Nebot, A., Castro, F., & Mugica, F. (2006). *Characterization of atypical virtual campus usage behavior through robust generative relevance analysis*. Retrieved from http://login.ezproxy.library.ualberta.ca/login?url=http://search.ebscohost.com/login.aspx?direct=true&db=edselc&AN=edselc.2-52.0-33847213198&site=eds-live&scope=site

von Davier, M. (1997). Bootstrapping goodness-of-fit statistics for sparse categorical data: Results of a Monte Carlo study. *Methods of Psychological Research Online, 2,* 29-48.

Wang, W., Weng, J., Su, J., & Tseng, S. (2004). Learning portfolio analysis and mining in SCORM compliant environment. In *ASEE/IEEE Frontiers in Education Conference* (pp. 17-24), Savannah, Georgia.

Witten, I. H., & Frank, E. (2005). *Data Mining: Practical Machine Learning Tools and Techniques, 2nd ed.* San Francisco, CA: Morgan Kaufmann.

Xiong, X., Zhao, S., Vaninwegen, E., & Beck J. (2016). Going deeper with deep knowledge tracing. In T. Barnes, M. Chi, & M. Feng (Eds.), *Proceedings of the 9th International Conference on Educational Data Mining* (pp. 666-668), Raleigh, North Carolina.

Zaïane, O., & Luo, J. (2001). Web usage mining for a better web-based learning environment. In *Proceedings of Conference on Advanced Technology for Education* (pp. 60-64), Vanff, Alberta.

Zapata-Rivera, D. & Bauer, M. (2011). Exploring the role of games in educational assessment. In M. C. Mayrath, J. Clarke-Midura, D. H. Robinson, & G. Schraw (Eds.), *Technology-*

based assessments for 21st century skills: theoretical and practical implications from modern research (pp. 149-172). Charlotte, NC: Information Age Publishing, Inc.

Zakrzewska, D. (2008). Using clustering technique for students' grouping in intelligent e-learning systems. In A. Holzinger (Ed.), *HCI and Usability for education and work: 4th symposium of the workgroup human-computer interaction and usability engineering of the Austrian computer society* (pp. 403-410). Berlin, Germany: Springer-Verlag.

Zeiler, M. D. (2012). "ADADELTA": an adaptive learning rate method. Retrieved from: http://www.matthewzeiler.com/pubs/googleTR2012/googleTR2012.pdf

Zhang, K., Cui, L., Wang, H., & Sui, Q. (2007). An improvement of matrix-based clustering method for grouping learners in e-learning. In *The 11th International Conference on Computer Supported Cooperative Work in Design* (pp. 1010-1015), Melbourne, Australia.

Zhang, Y., Shah, R., & Chi, M. (2016). Deep learning + Student modeling + Clustering, a recipe for effective automatic short answer grading. In T. Barnes, M. Chi, & M. Feng (Eds.), *Proceedings of the 9th International Conference on Educational Data Mining* (pp. 562-568), Raleigh, North Carolina.

Zhou, M., Xu, Y., Nesbit, J. C., & Winne, P. H. (2011). Sequential pattern analysis of learning logs: methodology and applications. In C. Romero, S. Ventura, M. Pechenizkiy, & R. S. J. d. Baker (Eds.), *Handbook of Educational Data Mining* (pp. 93-106). Boca Raton, FL: CRC Press.