STARS

University of Central Florida
STARS

Electronic Theses and Dissertations, 2004-2019

2019

Student Community Detection and Recommendation of Customized Paths to Reinforce Academic Success

Yuan Shao
University of Central Florida

 Part of the Computer Sciences Commons
Find similar works at: https://stars.library.ucf.edu/etd
University of Central Florida Libraries http://library.ucf.edu

This Masters Thesis (Open Access) is brought to you for free and open access by STARS. It has been accepted for inclusion in Electronic Theses and Dissertations, 2004-2019 by an authorized administrator of STARS. For more information, please contact STARS@ucf.edu.

STARS Citation

Shao, Yuan, "Student Community Detection and Recommendation of Customized Paths to Reinforce Academic Success" (2019). *Electronic Theses and Dissertations, 2004-2019*. 6304.
https://stars.library.ucf.edu/etd/6304

STUDENT COMMUNITY DETECTION AND RECOMMENDATION OF CUSTOMIZED PATHS TO REINFORCE ACADEMIC SUCCESS

by

YUAN SHAO
B.S. Wuhan University, 2005

A thesis submitted in partial fulfilment of the requirements
for the degree of Master of Science
in the Department of Computer Science
in the College of Engineering and Computer Science
at the University of Central Florida
Orlando, Florida

Spring Term
2019

© 2019 Yuan Shao

ABSTRACT

Educational Data Mining (EDM) is a research area that analyzes educational data and extracts interesting and unique information to address education issues. EDM implements computational methods to explore data for the purpose of studying questions related to educational achievements. A common task in an educational environment is the grouping of students and the identification of communities that have common features. Then, these communities of students may be studied by a course developer to build a personalized learning system, promote effective group learning, provide adaptive contents, etc. The objective of this thesis is to find an approach to detect student communities and analyze students who do well academically with particular sequences of classes in each community. Then, we compute one or more sequences of courses that a student in a community may pursue to higher their chances of obtaining good academic performance.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1: INTRODUCTION

One definition of Data Mining (DM) is that the process of extracting interesting, interpretable, useful and innovative information from data. It has been applied for businesses, scientists and governments in many years, to filter through volumes of data like online shopping records, weather data and census data [9]. EDM is the application of DM techniques to educational data, and its goal is to analyze these kinds of data so that one can address and resolve educational research issues.

One main research area of EDM is to group students, and we can use community detection approaches to group them. One of prevalent definitions of community detection is given by Newman and Girvan (2004): A community can be represented by a subgraph which has more edges within the subgraph than the rest of the graph. Likewise a graph has a clustering configuration if there are more links within any cluster than between those clusters [17].

A big amount of effort has been devote to develop community detection approaches, and there are two widely adopted thus worth to mention. The stochastic model Psorakis et al. [18] addresses on deriving generative models of networks. This kind of generative model uses an embedding in a low dimensional latent space to represent a network. Using a different approach, the modularity maximization model tries to maximize a modularity function on network frames. The Eigenvalue Decomposition (EVD) is applied to optimize the method, which is equal to rearranging a low-rank modularity matrix [12].

Machine learning is an emerging research area, and has applications on various fields. Some machine learning methods, such as classification and clustering, are commonly used to group students. Machine learning techniques can be divided into two kinds. One is supervised learning, where you have input variables and an output variable, and an algorithm is applied to learn the mapping function from the input to the output. The goal of supervised learning is to optimize the

mapping function, so that you can predict the output value when you get a new input value. In supervised learning the training data is labeled, so that the model can be optimized via reducing deviation of the predicted result and the labeled output variable during training, until it can achieve an satisfying accuracy of predication with new input data. Another kind of machine learning techniques is unsupervised learning, where you only have input variables and no output variable. The objective of unsupervised learning is to model the underlying structure in the data in order to learn more about the data. Unlike supervised learning, unsupervised learning deals with unlabeled data, and the algorithms learn to discover and present the interesting structure in the data.

The student data set used in this project is unlabeled, and the goal of this project is to build a model to find potential communities, which should be an general and universal model, and could be used also for any common university. So the unsupervised learning algorithms are preferable for this project.

Unsupervised learning techniques can be further divided into clustering and association algorithms. Clustering algorithms can group a set of observations in a manner that observations in the same cluster have more similarity to each other than to those in other clusters. Association algorithms aim to discover interesting relations between variables in large databases, to identify strong rules discovered in databases using some measures of interestingness. Like mentioned above, the goal of this project is to identify communities of students, so clustering algorithms are suitable for this task. There are two common clustering algorithms: K-means and hierarchical clustering, both have been used to experiment on a synthesis student data set.

In this work we have two steps, in Chapter 3 we have clustered students into subgroups based on their common characteristics. In Chapter 4 we have built a generative model to learn course sequences of high performance students in one subgroup, thereafter we can recommend new course sequences for new students, who belong to the subgroup.

CHAPTER 2: DATA SET

To guarantee the universality of our project's approach, that it can be applied on various universities and colleges, a synthesis data set with two files including undergraduate students information has been simulated for this project. One file includes administration information of 800 graduated undergraduate students with the same major: identifier number, first generation flag, transfer state(if started university directly), degree GPA. Another file has their course information, like course code number, title and taken term.

The administration data has been simulated based on a undergraduate student statistic of a state university [1]. Transfer student ratio has been set up to 14:86, which means 14% students in our data set are transferred from other institutions. First generation flag ratio has been set up to 1:4, that means 20% students in the data set are the first generation and other are not. We assumed a normal distribution of student's Degree GPAs: mean value is 3.2, standard deviation is 0.4 and range is from 0 to 4. After fixed the administration data distribution, we simulated 800 students' data randomly. For each student, an unique ordered identifier number was created, and his/her first generation flag, transfer state and degree GPA was generated randomly based on the pre-setting distribution.

Once student administration data was generated, student course data could be created. There are totally 8 courses in our data: "English Literature", "Composition", "Statistical Methods", "Physics Using Calculus I", "Physics Using Calculus II", "Principle Of Macro-economy", "Chemistry" and "Calculus". To apply 3 Markov Models to generate course sequences for those 800 students, we set transition and initial probabilities of those 3 models with following assumptions based on common sense: **Model I** is for high performance (who's degree GPA is higher than the average of their group) non first generation and non transfer students. The transition probability matrix and

initial probability vector of Model I is shown as Figure 2.1, we can see the initial probability of course "Composition" is 0.35, higher than other courses. As we assumed that these students would prefer to take course "Composition" than other courses firstly when they started at the university, to build a robust foundation of university-level education. Some courses are prerequisite of other courses, for instance, course "Calculus" is prerequisite to "Physics Using Calculus I", and course "Physics Using Calculus I" is prerequisite to "Physics Using Calculus II." Those relationships have been reflected in the matrix, for example: the transition probability from "Physics Using Calculus I" to "Physics Using Calculus II" has been set up to 0.75, much higher than any other courses. **Model II** is for the first generation high performance students (who's degree GPA is higher than the average of their group), the probability of taking math related fundamental courses (like "Calculus") at the beginning of their academic paths is higher than taking other courses such as "English Literature" and "Composition". Same like Model I, some courses are prerequisite of other courses, for example, course "Calculus" is prerequisite to "Physics Using Calculus I", and course "Physics Using Calculus I" is prerequisite to "Physics Using Calculus II." Transition and initial probabilities have been set up to reflect those assumptions, for instance, the transition probability from "Physics Using Calculus II" to "Physics Using Calculus I" is 0. The initial probability of "Calculus" of Model II is set to 46%, so that "Calculus" has much higher chance to show at first position of a sequence generated by Model II. **Model III** represents all other students who are not in cases of Model I and Model II. Its transition and initial probabilities have been set evenly.

```
mat1 <- matrix(nrow = 8, ncol = 8, 0, dimnames = list(c('English Literature',
        'Composition', 'Statistical Methods', 'Physics Using Calculus I',
        'Physics Using Calculus II', 'Principle Of Macroeconomy', 'Chemistry', 'Calculus'),
        c('English Literature', 'Composition', 'Statistical Methods', 'Physics Using Calculus I',
          'Physics Using Calculus II', 'Principle Of Macroeconomy', 'Chemistry', 'Calculus')))
mat1[1,] <- c(0, 0.15, 0.15, 0.15, 0.05, 0.2, 0.15, 0.15)
mat1[2,] <- c(0.45, 0, 0.1, 0.1, 0.05, 0.1, 0.1, 0.1)
mat1[3,] <- c(0.1, 0.1, 0, 0.25, 0.05, 0.3, 0.1, 0.1)
mat1[4,] <- c(0.05, 0.05, 0.05, 0, 0.75, 0.05, 0.05, 0)
mat1[5,] <- c(0.2, 0.2, 0.2, 0, 0, 0.2, 0.2, 0)
mat1[6,] <- c(0.25, 0.25, 0.2, 0.1, 0.1, 0, 0.05, 0.05)
mat1[7,] <- c(0.15, 0.15, 0.15, 0.2, 0, 0.2, 0, 0.15)
mat1[8,] <- c(0.1, 0.1, 0.2, 0.35, 0, 0.2, 0.05, 0)
vec1 <- c(0.15, 0.35, 0.1, 0, 0, 0.05, 0.1, 0.25)
```

Figure 2.1: Transition and initial probabilities for Model I

After set up those 3 Markov Models, we generated a course sequence for each student: chose model based on a student's administrative information. For instance, if the student was first generation and his/her GPA is equal or greater than 3.2, we would use Model II; If he/she is non first generation, non transfer and his/her GPA is equal or greater than 3.2, we would choose Model I; otherwise Model III would be applied. When a Markov Model generated a course sequence for a student, we also added course term code in order. Unique course number for each course has been added at end, then all sequences have been output to a csv file.

CHAPTER 3: CLUSTERING MODELS

Clustering is to divide population into subgroups, which are similar in the groups and different between groups. How to define similarity and difference needs domain knowledge, to choose related features.

Clustering method has applications in many fields, one instance is the market segmentation – to identify subgroups of people, who are more responsive to a specific form of advertising or who are more likely to buy a specific product [9]. Like the market segmentation, students can be grouped by their characters and backgrounds to subgroups of people, who are more likely to achieve academic success with particular course schedules. Different clustering algorithms have been implemented to group students, such as: hierarchical agglomerative clustering, K-means and Expectation-Maximization, to form heterogeneous groups based on student skills [16]. Among those algorithms hierarchical clustering and K-means clustering are two most known approaches, and both are based on dissimilarities inside and between groups.

Dissimilarity Matrices

Central to all clustering analyses is the definition of the similarity (or dissimilarity) degree between the individual objects being clustered. A clustering method tries to group objects according to the definition of similarity applied to it.

Sometimes data is expressed directly in terms of the proximity between pairs of objects. These can be either similarities or dissimilarities [7]. Most popular algorithms use a dissimilarity matrices as their input, and most common choice of calculating the dissimilarity between two objects i and $i^{'}$ for quantitative data is squared distance. However, the squared distance is not suitable for

categorical data. Furthermore, it's desirable to give each attribute different weight instead of equal weight to all attributes.

For mixed data with continuous and categorical features, Gower's general similarity measure is a good choice to calculate the similarity of mixed data types. Gower's distance can be used to measure how different two individuals are. Two objects i and j may be compared on a feature k then assigned a score zero when i and j are considered different; and a positive score, when they have some degree of similarity [11]. The records may contain combinations of logical, numerical, categorical or text data.

The first step of clustering is to calculate distances and produce a distance matrix. The second step is to use a clustering algorithm to group students. Two kinds of algorithms: Hierarchical Clustering and Partition Around Medoids(PAM) clustering have been used for this task.

Hierarchical Clustering

Hierarchical clustering is a common and developed algorithm in unsupervised machine learning. Hierarchical clustering approaches begin with splitting the data set into single nodes, then unite the nearest two data points into a new node till there is one last node left, which represents the whole data set. Many clustering approaches have this process in common, but how they measure the inter-cluster dissimilarity are different at each step. There are seven most common used methods: single, complete, average, weighted, Ward, centroid and median linkage [15].

A hierarchical clustering algorithm can generate a tree as graphical display of the result, called a dendrogram. This dendrogram illustrates the merging process and the subclusters. Similar data points and data subsets are close in a dendrogram. Figure 3.1 shows how four points can be merged into a single cluster [14]. Once a dendrogram is generated, we can get the wanted number

of clusters G by cutting the dendrogram at a height based on G branches [8].

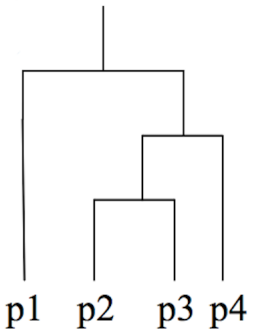

Figure 3.1: A dendrogram with four points

There are two major ways to implement a hierarchical clustering: One is agglomerative way, and it begins with all objects as singleton clusters, then merges the most similar pair of clusters. This demands a definition of cluster similarity. Another is divisive way, and it begins with one, all-inclusive cluster, then splits a cluster at each step, until only singleton clusters of individual objects remain. In this way, it requires to make decisions which cluster to split and how to split at each step [14]. Agglomerative approaches are more popular and have been applied in this paper.

PAM Clustering

One of the most fundamental and popular clustering techniques is K-means clustering algorithm. K-Means clustering algorithm uses a centroid to indicate a cluster. It splits a data set of n data points into k clusters, and each data point belongs to a cluster with nearest possible centroid. This method is interpretable and easy to implement, however it has a major disadvantage. When there are data points with extreme values, the data would distribute unevenly, causing inappropriate

clustering. Therefore K-Means clustering approaches are very sensitive to outliers and noise, this reduces its performance as well. K-means performs also not so well when clusters have non-convex shapes or very diverse sizes. So another clustering algorithm which is similar but still robust to outliers and noise is needed. Especially outliers and noise are common in real environment [6]. Partition Around Mediods (PAM) is one such algorithm. PAM was developed by Leonard Kaufman and Peter J. Rousseeuw, and it is very alike to K-means, as both are partitional algorithms. K-means and PAM both split a data set into groups, and both work by minimizing the error, though the latter uses Medoids. A medoid is a data point of a data set and represents a cluster, in which it is included. Unlike PAM, K-means uses Centroids. A centroid is an artificially generated item, which represents its cluster. Thus PAM is relatively robust than K-Means especially in the case of noise or outliers.

Experiments

A dissimilarity matrix with Gower's general similarity measure generated from the synthesis student data set which include administration information such as gender, first generation flag, transfer state and degree GPA, have been fed to two hierarchical clustering methods: complete links and average links. Figure 3.2 is the tree from average links, and Figure 3.3 shows the tree from complete links. There could be 2 or 4 clusters for both trees. Table 3.1 shows the comparison of labels between those two approaches when cut both trees with 2 clusters. Table 3.2 shows the comparison when cut both trees with 4 clusters. From the tables we can see, the two algorithms produced same labels and clusters. The dendrogram can show how many clusters you can get when cut it at different height, however there is no suggestion where to cut the tree and how many clusters should we choose. The instability of hierarchical clustering models requires domain knowledge and carefully choose parameters for good performance. A different approach is needed, which could suggest a

suitable cluster number.

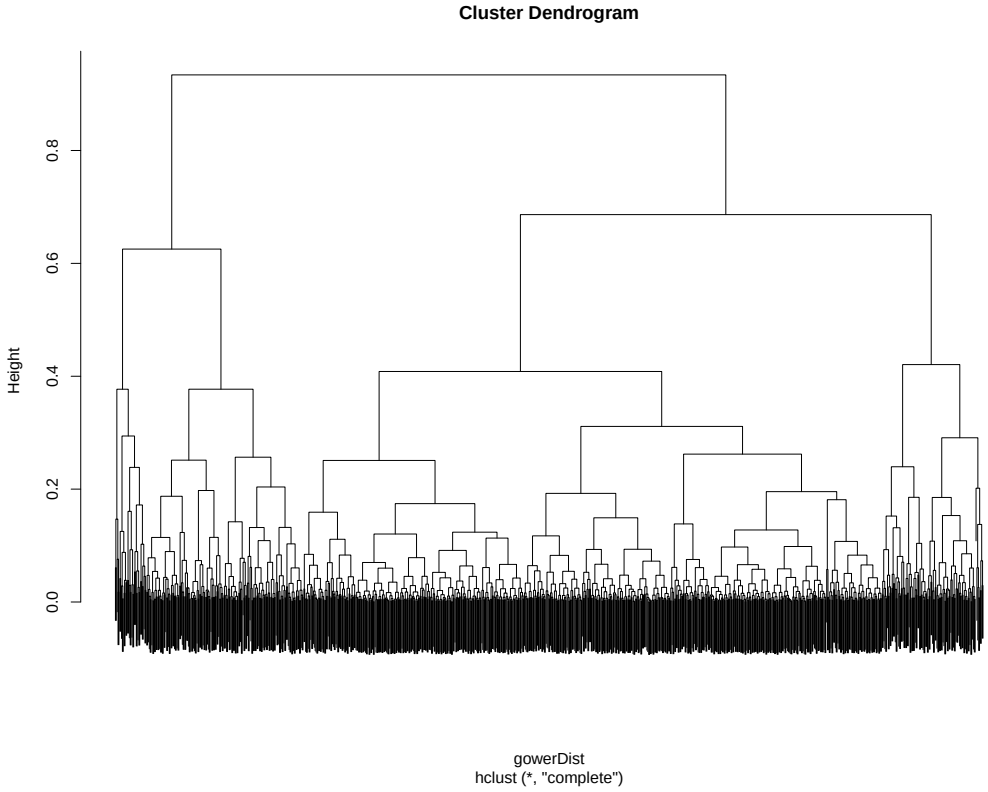

Figure 3.2: Students dendrogram with complete linkages

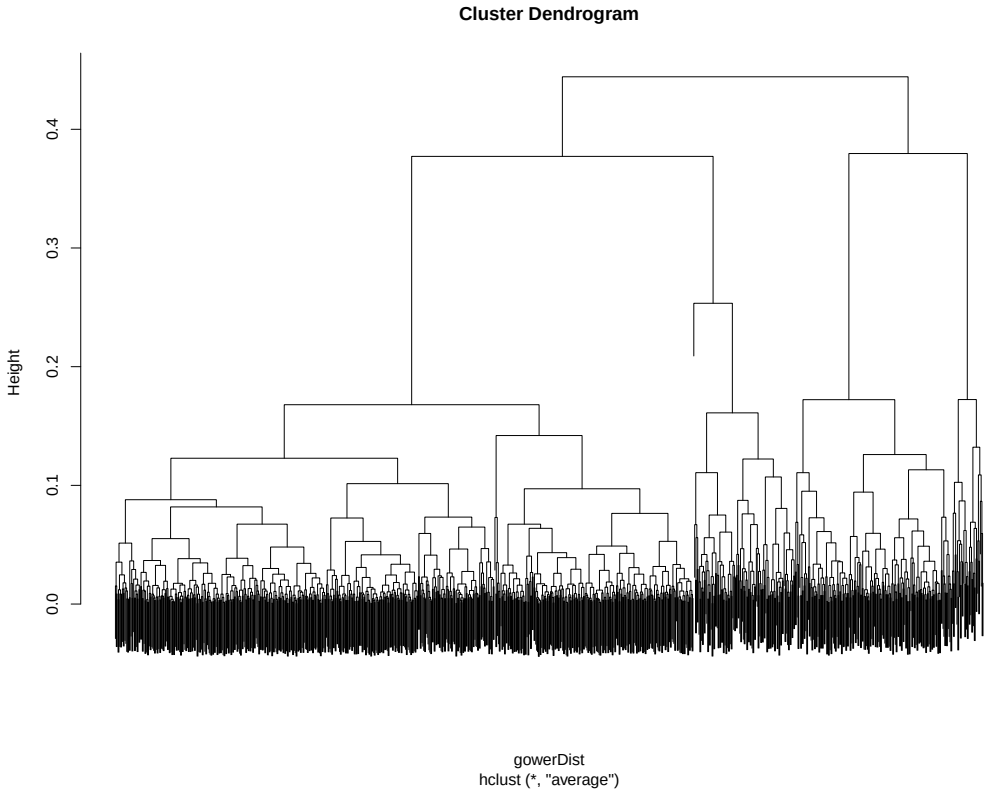

Figure 3.3: Students dendrogram with average linkages

		average links	
		1	2
complete links	1	783	0
	2	0	217

Table 3.1: Complete and average link 2 clusters comparison

		average links			
		1	2	3	4
complete links	1	666	0	0	0
	2	0	181	0	0
	3	0	0	36	0
	4	0	0	0	117

Table 3.2: Complete and average link 4 clusters comparison

The same dissimilarity matrix with Gower's general similarity measure has been applied for the PAM method, and silhouette width has been used to estimate the suitable K of clusters. Silhouette is a measurement of interpretation and validation of consistency within clusters of data. The measurement gives a graphical explanation of how well every object is within its group [20]. The silhouette value represents that how similar a data point is to its own group compared to the rest groups. The value's range is from -1 to +1, and a high value means, that the data point matches well to its own group and matches poorly to the rest groups. If most data points have high values, the clustering construct is proper. If Silhouette values of many points are low, the clustering construct may have too many or too few clusters [3].

From the Figure 3.4 we can see, silhouette width has the highest value when K is 3, which indicate 3 could be a good choice for K. As it achieves a high silhouette width value meanwhile keeps clustering relatively simple: not too many subgroups.

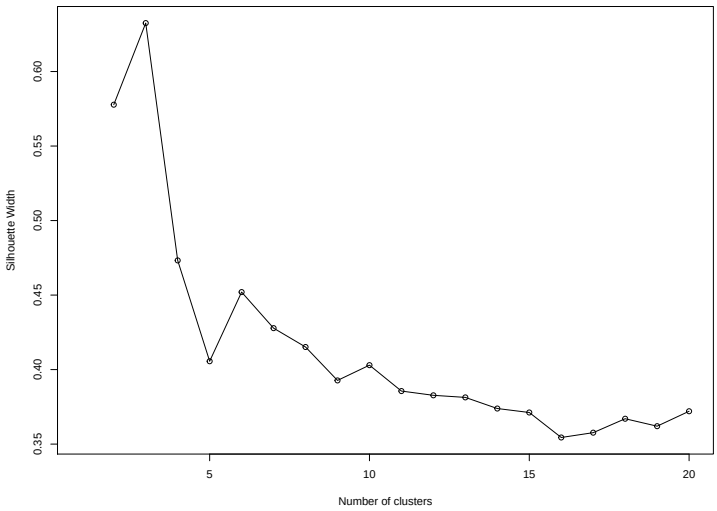

Figure 3.4: Silhouette width vs K

Then students have been divided into 3 subgroups, and Figure 3.5 shows a summary of one subgroup. We can see that two attributes 'First.Generation.Flag' and 'Begin.University.Directly' play important roles in the clustering, as all students in this subgroup are not first generation, and started to study in this university directly instead of transferring from other institutions. We chose this subgroup and divided it further: created a subset of students, whose degree GPA is above average, for our course sequence generator model. Course information of those students have been extracted, Figure 3.6 is an example of one student's course information, which shows what courses he/she took, at which terms and the course grades.

Above implementation has been done with R. See Appendix.

```
  Identifier    First.Generation.Flag Begin.University.Directly   Degree.GPA
Min.   :  1.0   0:666                 0:  0                      Min.   :2.070
1st Qu.:271.0   1:  0                 1:666                      1st Qu.:2.922
Median :513.5                                                    Median :3.220
Mean   :510.4                                                    Mean   :3.194
3rd Qu.:761.8                                                    3rd Qu.:3.460
Max.   :999.0                                                    Max.   :3.990
```

Figure 3.5: Student group summary

Term.Taken.Code	Course.Title	Course.Number
1000	Composition	1200
1100	Principle Of Macroeconomy	2001H
1200	Physics Using Calculus I	2000
1300	Physics Using Calculus II	2100
1400	Statistical Methods	1001A
1500	Chemistry	1560

Figure 3.6: Student course information sample

CHAPTER 4: GENERATIVE MODELS

Sequence Model Approaches

The end goal of this project is to compute one or more sequences of courses that a student in a community may pursue to increase their chances of obtaining good academic performance. It has some similarity with the sequence classification which is an common research task with many applications in different domains, such as e-commerce, computer security and finance [4]. The former aims to obtain patterns or event (course) distribution probability from training data and to generate event (course) sequences. The latter fits models to training data and predicts the class of a new given sequence by a discriminant or generative way; both handle event sequences. Therefore, our sequence generation approach has been derived from sequence classification.

The sequence classification methods can be broadly classified into three large groups. **Feature-based:** The methods in the first group represent each sequence as a d-dimensional feature vector, and use traditional supervised classifiers, such as naive Bayes, decision trees, support vector machines (SVM), or random forest for performing sequence classification. The main task of these methods is selecting suitable features that are effective for the classification task. **Distance-based:** The classification methods in this group design a distance/similarity function between a pair of sequences. Using this function, such a method partitions a set of input sequences into different classes. Kernel-based methods also fall into this group, as a kernel function is nothing but a similarity function between a pair of objects. **Model-based:** The third group consists of classification methods that are based on sequential models, such as Markov model, Hidden Markov Model (HMM), or graphical model, such as conditional random field (CRF). It also includes non-sequential probabilistic models, such as Naive Bayes [4].

Mode-based sequence classifiers match our case well, as they are generative models: A generative model for sequence classification assumes that sequences in each class c_i are being generated by a model Mc_i. Mc_i is defined over some alphabet Σ, and for any string s $\in\Sigma*$, Mc_i specifies the probability $P^{Mc_i}(s|c_i)$, which is the likelihood that the given sequence belongs to class i. A classifier can then be constructed as below. First, using a training data set, a probabilistic model Mc_i is trained for each class c_i, for i from 1 to k using the sequences belonging to the class c_i. Then, to predict an unlabeled sequence (say s) we simply use Bayes rule [4]:

$$class(s) = argmax_k \frac{P^{M_{Ci}(s|c_i)} \cdot P(c_i)}{\sum_{j=1}^{k} P^{M_{Cj}(s|c_j)} \cdot P(c_j)}$$

In our case we have been using two classes 'Excellent' and 'Normal' to represent students who have above average degree GPA and under average degree GPA. As the goal is to produce course-sequences of the 'Excellent' class, we only need to study the generative model of the 'Excellent' class. There are various sequence generative models, and Markov Model is a popular one which is applied in various fields, such as bioinformatics. Besides its generative nature, Markov Model has been chosen for our project because it's interpretive. This would be helpful for a course developer to understand how a model produces course-sequences.

Markov Model

Markov Model is a classical model that generates sequences in which the probability of a symbol depends on the previous symbol [19]. One of the most widely used models is a classical Markov Chain because of its simplicity on the one side and great ability to model various types of phenomena evolving in time on the other side. In the basic form we have a set of states and a process, which adopts just one state at every step. The probability of adopting a state at the next step only

relies on the state, that the process has at that time [21]. Common matrix computations are mostly applied to solve these problem.

Figure 4.1 is a simple Weather Markov Chain with 3 symbols (states) and arrows between states. Every arrow is related to a probability parameter, which defines the probability of a given state following another state. These parameters are named the transition probabilities, and we will use a_{st} to represent them:

$$a_{st} = P(\pi_i = t|\pi_{i-1} = s)$$

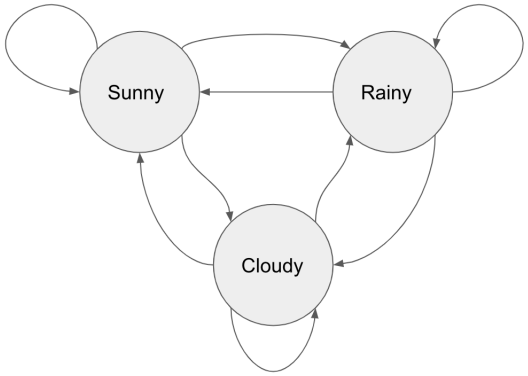

Figure 4.1: Markov chain example

By applying chain rule we can write the probability of a sequence for any probabilistic model of sequences as:

$$P(x) = P(x_L, x_{L-1}, ..., x_1) = P(x_L|x_{L-1}, ..., x_1)P(x_{L-1}|x_{L-2}, ..., x_1)...P(x_1)$$

A very important characteristic of a Markov chain is that the probability of every state x_i relies just on the preceding state x_{i-1}, instead of the all previous order, consequently: $P(x_i|x_{i-1}, ..., x_1) =$

$P(x_i|x_{i-1}) = a_{x_{i-1}x_i}$. The former equation can be transformed to:

$$P(x) = P(x_L|x_{L-1})P(x_{L-1}|, x_{L-2})...P(x_2|x_1)P(x_1) = P(x_1)\prod_{i=2}^{L} a_{x_{i-1}x_i}$$

Notice that as well as specifying the *transition probabilities* we must also give the probability $P(x_1)$ of starting in a particular state. Therefore we add a letter to the alphabet, which we will call B. By defining $x_0 = B$, we call the probability of the first letter in a sequence *initial probability*:

$$P(x_1 = s) = a_{Bs}$$

After formalized the notation for the Markov Model we illustrate how to calculate transition probabilities and the initial probability of a Markov model given sequences generated by the model. The probabilities are unknown, and we impose no restrictions on it meanwhile estimate it from data [2]. With the equation $P(x) = P(x_1)\prod_{i=2}^{L} a_{x_{i-1}x_i}$ and Maximum Likelihood Estimation, transition and initial probabilities can be calculated as below, where c_{st} is the number of times symbol t followed symbol s, and t' is any symbol which followed s.

$$a_{st} = \frac{c_{st}}{\sum_{t'} c_{st'}}$$

Experiments

After grouping students, the next step is to analyze the association between courses that students took and their GPAs. We assume that, besides which courses were taken by students, the sequences of courses taken are also important to their academic performance, because some courses are foundations of other advanced courses or could prepare students better with the necessary knowledge and skills for some interdisciplinary courses.

Input of a sequence model is generally an alphabet of events: E_1, E_2, E_3,... E_n. In our case, it can be represented as an ordered list of vectors. For instance, a student's course sequence can be shown as: $\{$(Calculus, Statistical Methods, Composition I, English Literature), (General Physics, Computer Science foundation, Algorithm), (Discrete Math I, Operating System, Object Oriented Programming)$\}$. Each vector shows the courses that a student took over a semester, and those vectors are in a time series. With those vectors we can derive all course sequences for all students, and use them as training data to fit our Markov model.

From the high-performance students group generated from the clustering experiment, course sequences have been derived based on the information of course-taken terms and course codes. Taking all courses would provide much noise information to models, therefore data have been filtered. As each student could take different courses, and they could take courses that are irrelevant to their majors based on their personal interests, so the 8 most frequently chosen courses have been chosen when we simulated the student course sequences. For each student, all combinations of course sequences with longest possible length (depending on how many terms he/she took) have been generated and stored. Figure 4.2 shows a few course sequence examples, as each line is one course sequence with different course code numbers. Every code number represents a course and those code numbers are in time series order.

```
[[204]]
[1] "1141H" "1200"  "2100"  "1100A" "1001A" "2000"

[[205]]
[1] "1100A" "1200"  "1100A" "2001H" "1001A" "1100A"

[[206]]
[1] "1200"  "1560"  "1001A" "1560"  "2000"  "2100"
```

Figure 4.2: Course sequence input sample

With the results mentioned in the previous section, we can calculate the transition probabilities and initial probabilities of our model. Course sequences have been fed into our Markov Model to produce a transition probability matrix and an initial probability vector. Once we have the transition probability matrix and the initial probability vector, we can produce a course recommendation with our model.

Figure 4.3 is an example of a 6 - courses sequence suggestion generated by the model, for a non first generation and non transfer student. Each number represents one course, followed its course title. We can see course 1200 "Composition" has been suggested at the first place, and it makes sense since the initial probability of course "Composition" is high(0.35, see Figure 2.1) when we simulated course data for non first generation, non transfer and high performance students with **Model I**. Our course recommendation model captured this feature from the course data. In the same way, course 2000 "Physics Using Calculus I," has been recommended ahead of 2100 "Physics Using Calculus II." The model learned the pattern that the former is a prerequisite of the latter, as we set this up in the transition probability matrix of **Model I**: the transition probability from "Physics Using Calculus I" to "Physics Using Calculus II" has been set up to 0.75, much higher than any other courses.

The implementation has been done with R. See Appendix.

```
[1] "1200"
[1] Composition
Levels: Composition
[1] "2001H"
[1] Principle Of Macroeconomy
Levels: Principle Of Macroeconomy
[1] "2000"
[1] Physics Using Calculus I
Levels: Physics Using Calculus I
[1] "2100"
[1] Physics Using Calculus II
Levels: Physics Using Calculus II
[1] "1100A"
[1] English Literature
Levels: English Literature
[1] "1001A"
[1] Statistical Methods
Levels: Statistical Methods
```

Figure 4.3: Course sequence suggestion sample

CHAPTER 5: CONCLUSION

Communities detection and recommendation for students are well developed research tasks with many effective solutions. However, what makes this paper different is that, we have applied both tools together to reinforce student's academic performance. We have created an approach to detect student communities and analyze students who do especially well academically with particular sequences of classes in each community, we then compute one or more optimal sequences of courses that a student in a community should pursue, to increase their chance to obtain good performance during college. We have shown how to implement this approach on a synthesis - student data set which simulated a university undergraduate students information. This approach can be used on universities and colleges in general.

CHAPTER 6: FUTURE WORK

Although a lot of extensions can be considered in this project, we indicate in continuation what arguably are the most interesting and influential among them. One is the Hidden Markov Model(HMM), which is an extension from Markov model. Y.S.Chen and H.S. Wong introduce a HMM based classification method to allow an e-learning system to rank various kinds of users by their navigation [5]. Their result demonstrates that the same method is capable of clustering student users to their relevant categories with high accuracy. Their project has similarity with our approach, so we assume HMM method has potential to develop our approach further. The HMM gets its name from two characteristics [10]. To begin with, it supposes that the result at time t was produced by some process whose state S_t is unknown for us. Furthermore, it also supposes that the state of this unknown procedure fulfills the Markov characteristic: given the value of S_{t-1}, the current state S_t is not dependent of all the states prior to t - 1. Comparing to the Markov Model, the HMM has state properties; for example, the transition matrix of student course generate model in our project could have two states: State1 and State2. The states could be interpreted according to the course developer's domain knowledge; for instance, they could be interpreted as 'novice' and 'experienced', showing how comfortable a student is with his or her major; or 'science' and 'liberal arts', which shows the focus of a student at that state. To build a HMM from a set of sequences, the Baum-Welch algorithm can been applied to generate transition and emission probability matrices for the HMM. The Baum-Welch approach is a penalized maximum likelihood method, which maximizing a posterior probability density over the model parameters [13]. Given student - course - sequence - transition probabilities between two states and emission probabilities, which indicate a course is chosen under a state, can be calculated, then we can produce course - sequence suggestions with the transition and emission probabilities.

Another approach worth to mention is the K-fold Markov chain. For this, a sequence is represented

by a graph in which every node means a sequence symbol, and every edge shows a dependency between two adjacent nodes. In other words, this model catches the dependency between the current symbol s_{k+1} and its k previous symbols $[s_{k+1}...s_1]$ in a sequence. The higher the value of k, the more complex the model is [4]. Like any other generative model, the probability can be estimated from data using the counts of the subsequences. Once all model parameters (probabilities) are known, a new sequence s can be assigned to the most likely class based on the generative model for each class.

At end we want to refer to large student data sets, collecting records in a large time range. With bigger data sets, more student attributes can be explored and the probability matrices of our course sequence generator can be closer to the reality.

APPENDIX A: R SOURCE CODE

```
#generate gower distance matrix
library(cluster)
gowerDist = daisy(adminData,metric = "gower",type = list(logratio = 3))

#hierarchical clustering algorithm with complete linkages
hcComplete = hclust(gowerDist)
plot(hcComplete,labels=FALSE)

#hierarchical clustering algorithm with average linkages
hcAverrage = hclust(gowerDist,method = 'average')
plot(hcAverrage,labels=FALSE)

#cut the dendrogram trees of complete and average linkages with 2 clusters
memberComp = cutree(hcComplete,2)
memberAver = cutree(hcAverrage,2)
table(memberComp,memberAver)

#cut the dendrogram trees of complete and average linkages with 4 clusters
memberComp = cutree(hcComplete,4)
memberAver = cutree(hcAverrage,4)
table(memberComp,memberAver)

#PAM algorithm
#calculate silhouette width as a function of k using PAM
silWidth = c(NA)
for(i in 2:20){
  pamFit = pam(gowerDist, diss = TRUE, k = i)
  silWidth[i] = pamFit$silinfo$avg.width
}

# Plot silhouette width (higher is better)
plot(1:20, silWidth, xlab = "Number of clusters",ylab = "Silhouette Width")
```

```
lines(1:20, silWidth)

# set k = 3, clustering interpretation via discreptive statistics
pamFit = pam(gowerDist, diss = TRUE, k = 3)

#get student groups by labels
group1 = adminData[pamFit\$clustering==1,]
group2 = adminData[pamFit\$clustering==2,]
group3 = adminData[pamFit\$clustering==3,]

#use group 1 to analyze course sequence and select subset with GPA above
    average from it
summary(group1)
mean(group1$Degree.GPA)
highPerfor = group1[group1$Degree.GPA$>$mean(group1$Degree.GPA),]

#add course info
highPerfCour = merge(highPerfor,course,by="Identifier")

#generate course list of high performance students
id = unique(highPerfCour$Identifier)
highPerfCour = highPerfCour[order(highPerfCour$Term.Taken.Code),]
highPerfCour = highPerfCour[order(highPerfCour$Identifier),]
courseList = list()
for (i in id){
  df = highPerfCour[highPerfCour$Identifier==i,c('Term.Taken.Code','Course.
      Number')]
  term.taken = as.list(unique(as.factor(df$Term.Taken.Code)))
  tempList = list()
  for (t in term.taken){
    te = as.character(df[which(df$Term.Taken.Code==t),'Course.Number'])
    tempList[length(tempList)+1] = list(te)
```

```
  }
  courseList[length(courseList)+1] = list(tempList)
}

#create training sequences
courseSequence <- list()
for (i in 1:length(courseList)){
  tList <- courseList[[i]]
  v = vector()
  for (j in 1:length(tList)){
    v[j] = as.vector(tList[j][1])
  }
  seq = do.call(expand.grid,v)
  for (i in 1:nrow(seq)){
    ch <- character()
    for (j in seq[i,]){
      ch[length(ch)+1] <- as.character(j)
    }
    courseSequence[length(courseSequence)+1] = list(ch)
  }
}

#Markov model
#generate transition prob matrix and initial probilities
topCourseNumber = droplevels(unique(highPerfCour$Course.Number))
probMat = matrix(nrow = length(topCourseNumber),ncol = length(topCourseNumber)
    ,0)
rownames(probMat) = topCourseNumber
colnames(probMat) = topCourseNumber
initialProb = rep(0,length(topCourseNumber))
names(initialProb) = topCourseNumber
for (i in 1:length(courseSequence)){
```

```
  x <- as.vector(courseSequence[[i]])
  initialProb[x[1]] = initialProb[x[1]] + 1
  if(length(x)>1)
  {for (i in 1:(length(x)-1))
  probMat[x[i], x[i+1]] = probMat[x[i], x[i+1]] + 1
  }
}

#get transition probabilities
for (i in 1:length(topCourseNumber)) probMat[i, ] <- probMat[i, ] / sum(
    probMat[i, ])

#generate initial probabilities
initSum <- sum(initialProb)
for (i in 1:length(topCourseNumber)) initialProb[i] <- initialProb[i]/initSum

#build a function to generate a sequence with the matrix
courseSeqGenerator = function(courses,seqLen,probMat,initialProb){
  newSequence = character()
  firstCourse = sample(courses,1,rep=T,prob = initialProb)
  newSequence[1] = as.character(firstCourse)
  for(i in 2:seqLen){
    preCourse = newSequence[i-1]
    probabilities = probMat[preCourse,]
    newCourse = sample(courses,1,rep=F,prob = probabilities)
    newSequence[i] = as.character(newCourse)
  }
  return (newSequence)
}

#call cour seSeqGenerator() function and create a new 6 course sequence
    suggestion
```

```
courseSuggestion = courseSeqGenerator(topCourseNumber,6,probMat,initialProb)
for (i in courseSuggestion){
  print(i)
  print(droplevels(unique(highPerfCour$Course.Title[highPerfCour$Course.Number
      ==i])))
}
```

LIST OF REFERENCES

[1] College transfer statistics. Retrieved from http://www.collegetransfer.net/UniversityOfCentralFlorida/TransferProfile/tabid/145/Default.aspx.

[2] Maximum likelihood estimation for markov chains. Retrieved from https://www.stat.cmu.edu/~cshalizi/462/lectures/06/markov-mle.pdf.

[3] Silhouettes(clustering). Retrieved from https://en.wikipedia.org/wiki/Silhouette_(clustering).

[4] Charu C. Aggarwal. Data Classification Algorithms and Applications. 2014.

[5] Y.S. Chen A.W.P.Fok, H.S. Wong. Hidden markov model based characterization of content access patterns in an e-learning environment. 2005.

[6] Aruna Bhat. K-medoids clustering using partitioning around medoids for performing face recognition. 2014.

[7] Sebastian Ventura Cristbal Romero. Educational data mining: A review of the state-of-the-art. 2010.

[8] Nema Dean Elizabeth Ayers, Rebecca Nugent. A comparison of student skill knowledge estimates. 2009.

[9] Trevor Hastie Gareth James, Daniela Witten and Robert Tibshirani. Introduction to statistical learning with r.

[10] Zoubin Ghahramani. An introduction to hidden markov models and bayesian networks. *IJPRAI*, 2001.

[11] J. C. Gower. A general coefficient of similarity and some of its properties. 1971.

[12] Dongxiao He Chuan Wang Xiao Wang Weixiong Zhan Liang Yang, Xiaochun Cao. Modularity based community detection with deep learning. 2011.

[13] David J.C. MacKay. Ensemble learning for hidden markov models. 1997.

[14] Vipin Kumar Michael Steinbach, George Karypis. A comparison of document clustering techniques. 2000.

[15] D. Müllner. Modern hierarchical, agglomerative clustering algorithms. September 2011.

[16] N. Myller, J. Suhonen, and E. Sutinen. Using data mining for improving web-based course design. 2002.

[17] M. E. J. Newman and M. Girvan. Finding and evaluating community structure in networks. 2004.

[18] Ioannis Psorakis, Stephen Roberts, Mark Ebden, and Ben Sheldon. Overlapping community detection using bayesian non-negative matrix factorization. 2011.

[19] Anders Krogh Richard Durbin, Sean R. Eddy. Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids. 1998.

[20] Peter J. ROUSSEEUW. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. 1986.

[21] Damjan Skulj. Discrete time markov chains with interval probabilities. 2009.