Aalto University
School of Science
Degree Programme in Computer Science and Engineering

Yang Zhao

Spot Anomalies from Patient Visits Data

Master's Thesis
Espoo, June 18, 2011

DRAFT! — September 8, 2016 — DRAFT!

Supervisors:	Professor Antti Ylä-Jääski, Aalto University
	Professor Pekka Perustieteilijä, University of Helsinki
Advisor:	Olli Ohjaaja M.Sc. (Tech.)

Aalto University
School of Science
Degree Programme in Computer Science and Engineering

ABSTRACT OF
MASTER'S THESIS

Author:	Yang Zhao		
Title:			
Spot Anomalies from Patient Visits Data			
Date:	June 18, 2011	**Pages:**	54
Major:	Data Communication Software	**Code:**	T-110
Supervisors:	Professor Antti Ylä-Jääski Professor Pekka Perustieteilijä		
Advisor:	Olli Ohjaaja M.Sc. (Tech.)		

A dissertation or thesis is a document submitted in support of candidature for a degree or professional qualification presenting the author's research and findings. In some countries/universities, the word thesis or a cognate is used as part of a bachelor's or master's course, while dissertation is normally applied to a doctorate, whilst, in others, the reverse is true.

!FIXME **Abstract text goes here (and this is an example how to use fixme).** FIXME! Fixme is a command that helps you identify parts of your thesis that still require some work. When compiled in the custom `mydraft` mode, text parts tagged with fixmes are shown in bold and with fixme tags around them. When compiled in normal mode, the fixme-tagged text is shown normally (without special formatting). The draft mode also causes the "Draft" text to appear on the front page, alongside with the document compilation date. The custom `mydraft` mode is selected by the `mydraft` option given for the package `aalto-thesis`, near the top of the `thesis-example.tex` file.

The thesis example file (`thesis-example.tex`), all the chapter content files (`1introduction.tex` and so on), and the Aalto style file (`aalto-thesis.sty`) are commented with explanations on how the Aalto thesis works. The files also contain some examples on how to customize various details of the thesis layout, and of course the example text works as an example in itself. Please read the comments and the example text; that should get you well on your way!

Keywords: ocean, sea, marine, ocean mammal, marine mammal, whales, cetaceans, dolphins, porpoises

Language: English

Acknowledgements

I wish to thank all students who use LaTeX for formatting their theses, because theses formatted with LaTeX are just so nice.

Thank you, and keep up the good work!

Espoo, June 18, 2011

Yang Zhao

Contents

Chapter 1

Introduction

1.1 Why Anomalies Matter?

As the elderly population increases, demand for outpatient services rises, which increases the operation cost in hospital [9][13]. This phenomenon requires improvements on the efficiency in healthcare service institutes. Among several potential aspects on improving efficiency, one expection is to make the whole visit more smoothly.

A patient visit to the hospital consists of several phases. From enrolling to doctor treatment, each phase are affected by many factors which could result in an unplesant experience. Examples are long waiting time, disordered treatment procedures etc. To help provide better visit experience, it is important to find out unexpected careflows. From studying these abnormal cases, administrative stuffs can understand the reasons for causing the problems. Thus, the stuffs can balance resources allocated in the hospital for smoother service in the future. What is better, if real-time anomaly detection is available, then the stuffs can provide necessary help to the patient in time.

1.2 Background: State Flow of the X-Akseli System

This thesis studies the anomaly detection problem using data extracted from Oulu University Hospital. The hospital uses X-Akseli queue system which aims at making patient reception fluent and effortless. This section describes how the X-Akseli reservation system works, in order to give the reader a general idea about how the data was produced. Patience privacy has been

Table 1.1: One example visit to the hospital.

reservationid	eventname	time
21332189	ENROLLING	2014-03-26 08:02:42.353
21332189	ARRIVED_IN_HOSPITAL	2014-03-26 08:02:42.517
21332189	WAITING	2014-03-26 08:03:29.007
21332189	CALLED	2014-03-26 08:07:15.061
21332189	IN_TREATMENT_ROOM	2014-03-26 08:07:15.072
21332189	CLOSED	2014-03-26 08:13:11.002

protected by using an anonymous ids. The reservation system has gone through several minor updates while recording these data. Considering this situation, the discussion in this thesis adhere to the latest system, version 1.18.3.

Assuming a patient has already made a reservation online. A typical visit scenario to the hospital is as follow. The patient arrives at the hospital lobby. Then the patient takes a queue number at the self-service kiosk, with the information showing in which area the patient should wait. Next, the patient goes to the correct waiting area, shows the queue number to another kiosk in the area to check in. After this, the patient can sit down and wait to be called by the doctor. When the doctor is available, the doctor will call the patient, treat the patient and close the whole visit. If assisted diagnose is needed, the doctor may pause the treatment. After other diagnoses are finished, the docotr then continues the treatment. In small departments, there may be no seperatation of waiting area and lobby. In this case, the first two steps will be integrated. The patient will not have to show the queue to another kiosk. However, there will still be two events recorded in the backend system, but with zero transition time. The whole procedure is shown in Figure 1.1 and the state flow of the backend system is shown in Figure 1.2

One example visit is listed in Table 1.1. Notice that, there are 6 events in this visit. However, the first two events `ENROLLING` and `ARRIVED_IN_HOSPITAL` happened in less than 1 second. This also happens to the 4th and 5th events, `CALLED` and `IN_TREATMENT_ROOM`. It can be considered that the two events in these two pairs happen simutaneously. To reduce redundant information, in this thesis, only following 7 events are considered. They are: `ENROLLING`, `WAITING`, `IN_TREATMENT_ROOM`, `PAUSED`, `IN_TREATMENT_ROOM_FROM_PAUSED`, `CLOSED`, and `CANCELLED`.

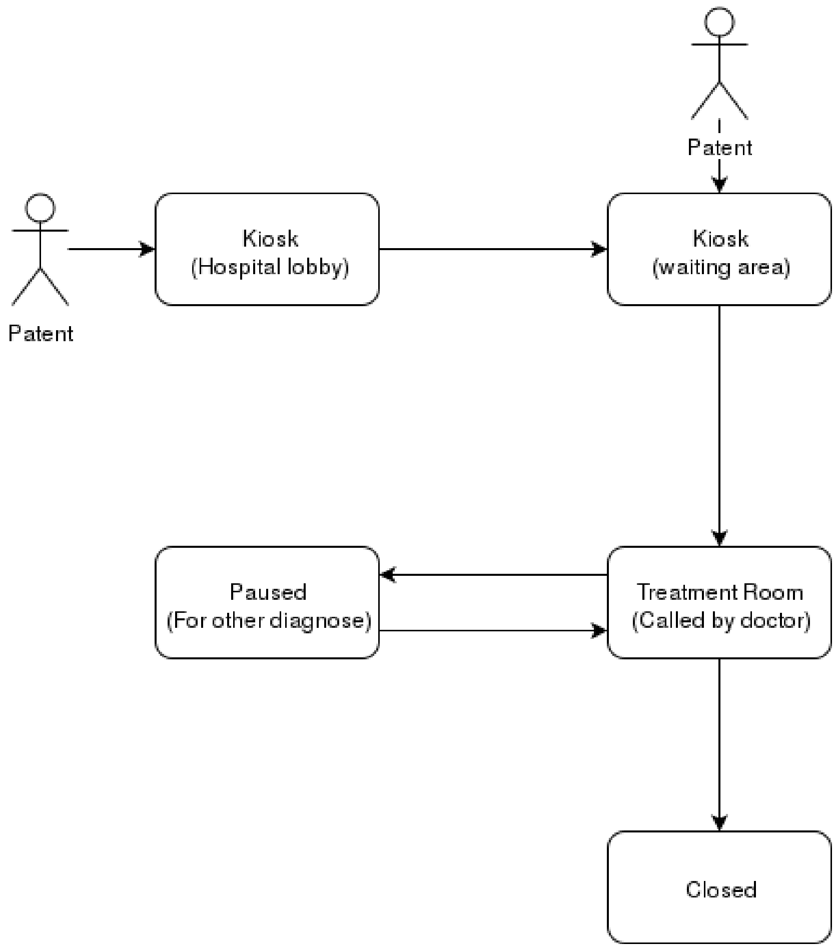

Figure 1.1: A typical visit scenario in hospital.

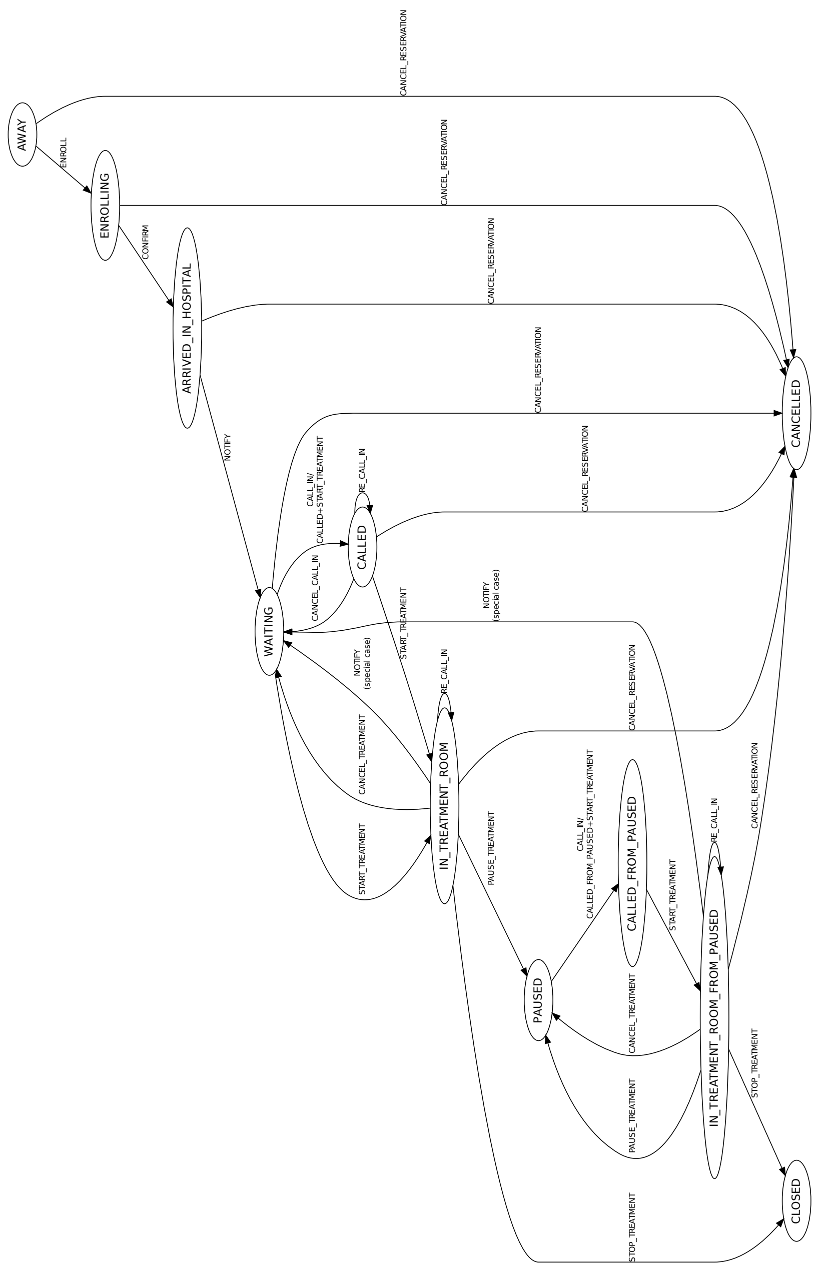

Figure 1.2: State flow in X-Akseli system, version 1.18.3.

1.3 Problem Description

Spotting an anomaly from hospital visits can be seen as finding outliers in a database containing time series entries. The assumption is that, the number of anomalies in the database is quite few compared to the number of normal entries. One naive approach to solve this problem is extreme value detection [1]. The idea is to set a threshold and classify any entry which has a value exceeds the threshould to be an anomaly. In the patient visits case, the threshould can be set to be a specific waiting time. Any visits have a longer time than the threshould is an anomaly. However, this approach have one obvious drawback due to the hard assignment. Selection of the threshould value determines the performance of this method. However, the boundary between anomaly and normal data is not clear. There could be a "gray area" around this threshould value. Additionally, definition of anomalies in patient visit is not limited to durations. Anomalies may also consists of strange behavior of the patient, such as being in `WAITING` states for several times. This may be caused by going to wrong waiting areas.

An alternative approach is to scoring each entry by showing how likely it could be an anomaly [10]. Then, anomalies are detected based on these scores. Typical methods include clustering methods and Markovian models. This approach considers anomalies from overview aspect. The concept is that, a single minor misbehavior doesn't lead to an anomaly. It is a sequence of uncommon behaviors that results in anomalies. This thesis decides to explore the problem using this approach.

1.4 Thesis Structure

The thesis aims at suggesting a practical method for detecting anomalies in hospital visits. The strucutre of this thesis is organized as follow. Chapter 2 introduces potential clustering method. Chapter 3 introduces potential generative model methods. Then Chapter 4 first describes more details and pre-process excecuted on the data used in the experiment. Then this chapter compares strength and drawbacks of these methods, discusses their applicability, and describes the experiment setup. Finally, Chapter 5 presents and analyzes the obtained results.

Chapter 2

Clustering Patient Visits

As described in Chapter XX, all patients data is retrieved from the database directly without any manual labeling. Due to lack of labels, unsupervised learning algorithms should be adopted. Clustering is a collection of unsupervised methods, which identifies groups of data points according to a defined similarity metric, such that objects in the same group posses higher similarities compared to objects in other groups. The clustering process does not rely on labels but the choice of similarity metrics. Variations in similarity metrics lead to different clustering methods.

Applying clustering methods in anomaly detection tasks has been studied numerously [11] . This chapter introduces two typical methods, K-Means [16] and DBSCAN [7]. Problem formulation, solutions, and potential issues are formally described using elaborated notations in following sections. However, analysis on the performance and constraints of these two methods are postponed to Chapter XXX, which reveals their practicality and infeasibility in the previously described anomaly detection problem. (Time, Memory consumption. For K-Means, definition of center.)

2.1 K-Means

K-Means is one of the simplest unsupervised algorithm which solves clustering problem. Despite its simplicity, K-Means has gained success in various situations, including anomaly detection [11] [4], image segmentation and compression [8], and preprocessing for more complicated algorithms. The method can be formally defined as follow: Given a data set $\{\mathbf{x}_1, ..., \mathbf{x}_N\}$ consisting of N observations in D-dimensional space, the object is to partition the data into K groups, by defining a set of K centers $\{\boldsymbol{\mu}_1, ..., \boldsymbol{\mu}_k\}$ in the same space, and assigning each observation to exactly one center point. Each center point represents a prototype associated with the k^{th} cluster.

The assignments can be represented using 1-of-K scheme. Then, for each data point x_n, a corresponding K-dimensional variable consisting of K binary elements $r_{nk} \in \{0, 1\}$ is introduced. Among these r_{nk}, exactly one of them equals 1, which means $\mathbf{x}_n$ belongs to the k^{th} cluster. Using this notation, evaluation of the clustering quality can be defined using the object function as follow:

$$J = \sum_{n=1}^{N} \sum_{k=1}^{K} r_{nk} D(\mathbf{x}_N - \boldsymbol{\mu}_k) \tag{2.1}$$

where D is the dissimilarity metric. Common choice of the metric is l_1-norm or l_2-norm. Mahalanobis distance is also adopted while considering the covariances between the K-dimensions [5]. In the following context, l_2-norm is adoped for discussion. Intuitively, this function can be considered as the distance summation of each point to its corresponding cluster prototype $\boldsymbol{\mu}_k$. The K-Means aims at finding a set of $\boldsymbol{\mu}_k$ which minimizes the object function. Finding the optimal solution for the above object function proves to be NP-Hard [2]. However, employing heuristic algorithms enables finding converged local optimal solutions. Section 2.1.1 describes one iterative algorithm, EM. Section 2.1.2 explores common issues related to K-Means and remedies.

2.1.1 EM Algorithm in K-Means

EM algorithm is an iterative algorithm to find local maximum. Each iteration consists of two phases, Expectation and Maximization, which corresponds to minimize the objective function J with respect to r_{nk} and $\boldsymbol{\mu}_k$ respectively. In the E(expectation) step, the algorithm minimize J with respect to r_{nk}, while keeping the $\boldsymbol{\mu}_k$ fixed. Then in the following M(maximization) step, the algorithm minimizes J with respect to $\boldsymbol{\mu}_k$, while keeping r_{nk} fixed.

Considering the optimization in E step, a critical observation of (2.1) is that $\mathbf{r}_n$'s are independent of each other. Thus, optimization on $\mathbf{r}_n$'s can be done seperately for each n. The solution is simply setting the r_{nk} corresponding to the minimum $\|\mathbf{x}_n - \boldsymbol{\mu}_k\|^2$ to 1. Intuitively, the algorithm assigns $\mathbf{x}_n$ to the closest cluster center. Formally, the solution can be written as

$$r_{nk} = \begin{cases} 1 & \text{if } k = \arg\min_j \parallel \mathbf{x}_n - \mu_j \|^2 \\ 0 & \text{otherwise} \end{cases} \tag{2.2}$$

In the following M step, the above determined r_{nk} is clamped. Then, $\boldsymbol{\mu}_k$ appears only in a quadratic term in J. Setting derivatives of J with respect to $\boldsymbol{\mu}_k$ to zero, solution formula for $\boldsymbol{\mu}_k$ can be expressed in following closed form

$$\boldsymbol{\mu}_k = \frac{\sum_n r_{nk}\mathbf{x}_n}{\sum_n r_{nk}} \tag{2.3}$$

This step can be considered as recomputing the cluster prototype by setting $\boldsymbol{\mu}_k$ to the mean of all points assigned to that cluster.

EM keeps executing these two steps alternatively, until a convergence happens or until number of iterations exceeded a predifined value. Since in each phase, one variable is fixed and updating another variable minimizes the cost function J, convergence is guaranteed. Formal proof on convergence has been studied by MacQueen [18].

The algorithm is illustrated using dataset generated independently from three gaussian distributions in Figure 2.1. In this demonstration, the algorithm takes $K = 3$, which is the correct number of components. Before running the first iteration, initialized $\boldsymbol{\mu}_k$ is required. This initialization is done by choosing three objects from the data set randomly.

2.1.2 Issues in K-Means

Despite the simplicity of K-Means, several underlying issues exists. The first potential is that, how to choose the value for K. In the above illustration, K was set to 3 which is the correct number of components. However, if K wasn't set to the correct value, unsatisfied clustering may be generated. Example of this issue is shown in Figure 2.2(a)-(c). To solve this problem, one practical way is drawing graph of the cost function versus value of K, as shown in Figure 2.2(d). Intuitively, when K is under smaller than the true number of clusters, increasing K will lead to a huge drop of cost function value. However, when K has reached or exceeded the correct value, incresing K leads only small cost function value drop. Thus, number of clusters corresponding to the 'elbow' point can be considered as the real number of clusters.

Another issues of K-Means relates to the initialization of $\boldsymbol{\mu}_k$. Since EM finds only local optimal solutions, a poor initialization could lead to worse clustering result. To avoid this problem, it is practical to run K-Means for several times and choose the best result according to the value of the cost function.

One more critical issue of K-Means lies in the choice of similarity metric. As mentioned at the beginning of this chapter, variations in similarity metric leads to different clustering methods. Choosing l-2 norm is convenient in terms of computation, but this choice limits the type of data variables to certain types. Using l-2 norm on categorical data is inappropriate since no ordering of categorical values exists. Also, this makes computing the mean

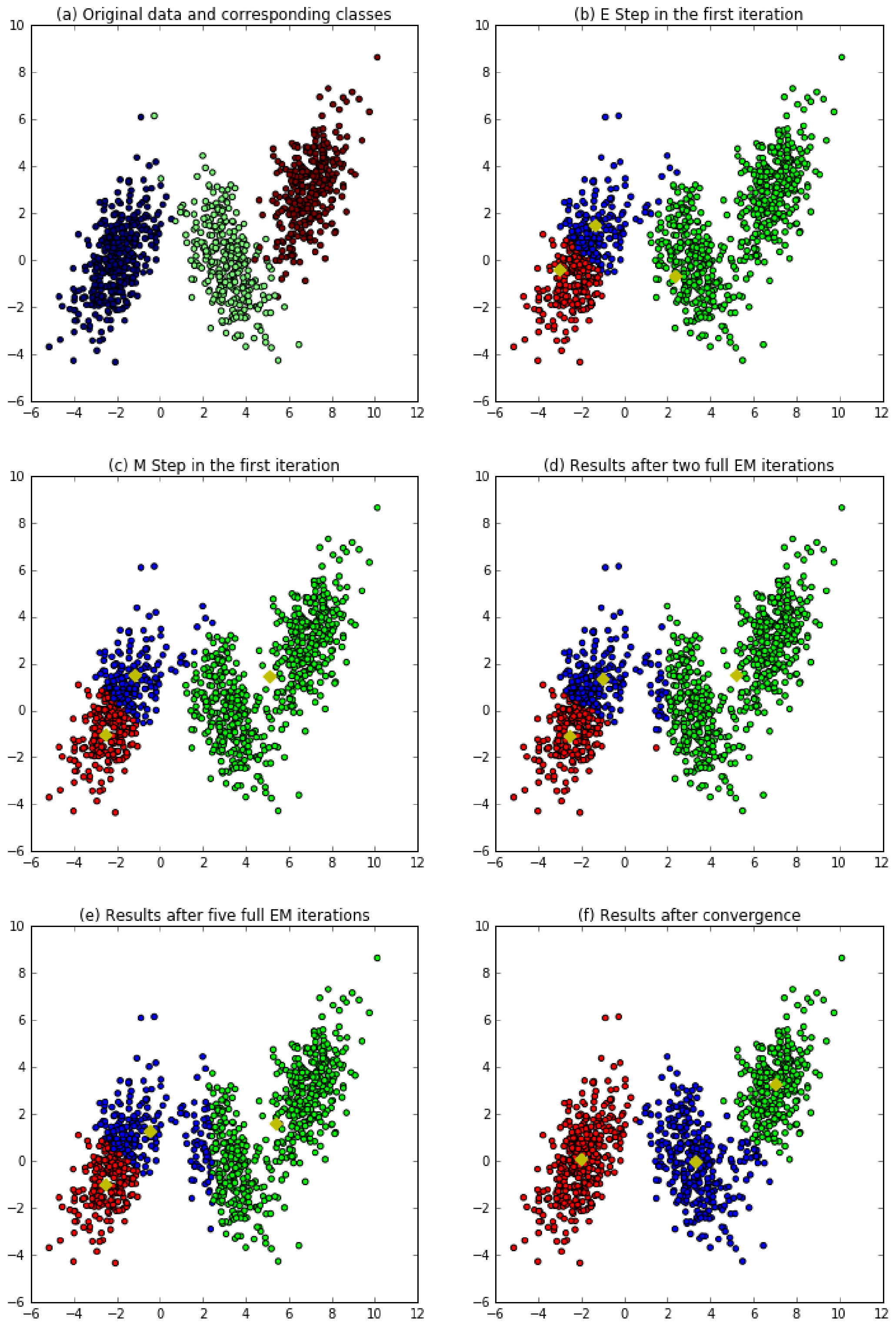

Figure 2.1: Illustraion of EM algorithm on K-Means using data generated independently from three gaussian distribution. (a) Original data and corresponding classes. Classes are denoted in different colors. (b) Assignments of each data after the first E step. The yellow diamonds represent the intialized $\boldsymbol{\mu}_k$. (c) Updated $\boldsymbol{\mu}_k$ after the M step in the first iteration. (d)-(f) Clustering results after several successive full EM iterations until convergence is met.

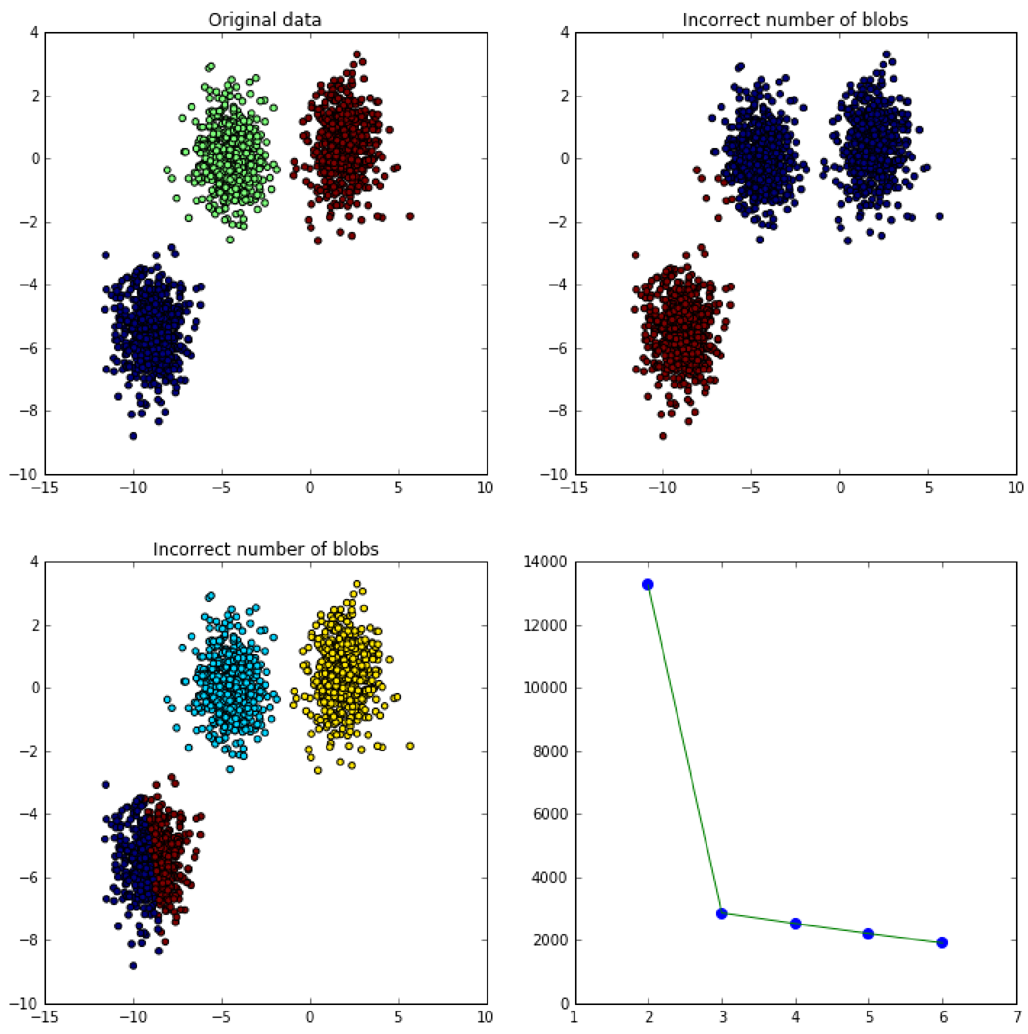

Figure 2.2: Issues of K-Means incurred by choosing inappropriate value for K

value a hard problem. To use K-Means on other data types, the similarity metric should be elaborately designed.

Besides, K-Means tends to form clusters into a convex space. As shown in Figure 2.1(b)-(e), the boundary between two different clusters forms a line lying at the midway and is perpendicular to the line connecting two cluster prototypes. However, a cluster is not necessary to be convex. This also limits the application of K-Means.

Finally, K-Means is also sensitive to noises. As shown in formula 2.3, updating $\boldsymbol{\mu}_k$ involves computing the mean of all data points assigned to that cluster. When noise objects with large deviation from other points in this cluster exist, update of $\boldsymbol{\mu}_k$ will be strongly affected.

2.2 Density Based Clustering Methods

This section explores another type of clustering method which forms cluster from the view of density aspect. **D**ensity-**B**ased **S**patial **C**lustering of **A**pplications with **N**oise (DBSCAN) [7] considers to be one of the most successful method from this category [15]. Compared to K-Means, DBSCAN posseses following advantages: 1). DBSCAN can detect clusters of arbitrary shape. 2). DBSCAN can determine the number of clusters automatically. 3). DBSCAN is robust to noises.

The following of this chapter is organized as follows. Section 2.2.1 compares and reveals the relation of DBSCAN to a basic graph traversal algorithm **D**epth **F**irst **S**earch(DFS). This section will introduce the critical concept used by DBSCAN which differentiate it with DFS. Section 2.2.2 describe this concept in more details. Section 2.2.3 compares DBSCAN and K-Means, and discusses practical issues of DBSCAN.

2.2.1 DBSCAN and DFS

Before introducing DBSCAN, it would be helpful to review a basic graph traversal algorithm, DFS, which is an algorithm which traverses or visits a graph. In the discussion of this context, the whole process is divided into two procedures DFS and DFS-Visit. A typical recursive implementation is described as follow.

```
DFS($G$)
1  for each vertex $u \in G$
2      $u.visited$ = FALSE
3  for each vertex $u \in G$
4      if $u.visited$ == FALSE
5          DFS-Visit($G, u$)
```

```
DFS-Visit($G, u$)
1  $u.visited$ = TRUE
2  for each vertex $v \in G.Adj[u]$
3      if $v.visited$ == FALSE
4          DFS-Visit($G, v$)
```

The first two lines in DFS initializes the algorithm by setting all nodes to be unvisited. Then, the algorithm traverse vertices in the graph. Once an unvisited node is found, DFS-Visit is called. In the procedure DFS-Visit, the given node u is labled as visited. If an unvisited child v of u is found, this procedure goes one layer deeper by calling another DFS-Visit on v. After finishing DFS-Visit(G, u), the algorithm backtrack to visit other unvisited children of u, which are siblings of v.

Assume DFS executes on an undirected graph. Each call of DFS-Visit on an unvisited node u explores u and all nodes reachable to it. These nodes form a component which disconnects with other components by other runs of DFS-Visit. Thus, each component can be considered as a cluster. The criterion to form a cluster is that each pair of nodes can reache each other through a path consisting of nodes only in this cluster.

DBSCAN can be seen as an application of DFS. However, DBSCAN differs from standard DFS from its useage of DFS-Visit. In standard DFS, the algorithm goes deeper by calling DFS-Visit(G, v) on all unvisited children of node u. In DBSCAN, however, the algorithm goes deeper if and only if node u satisfies extra conditions, which are specified by two user given value *Eps* and *MinPts*. These extra conditions make the component generated from DBSCAN a meaningful cluster viewing from the density side. The pseudocode is shown below.

```
DBSCAN($S, Eps, MinPts$)
 1  $ClusterId = 1$
 2  for each point $u \in S$
 3      if $u.label \neq$ NIL
 4          continue
 5      $u.neighbor =$ REGION-QUERY($u, Eps$)
 6      if $\|u.neighbor\| < MinPts$
 7          $u.label =$ NOISE
 8      else
 9          EXPAND-CLUSTER($u, ClusterId, Eps, MinPts$)
10          $ClusterId = ClusterId + 1$
```

```
EXPAND-CLUSTER($u, ClusterId, Eps, MinPts$)
1  $u.label = ClusterId$
2  for each point $v \in u.neighbor$
3      if $v.label ==$ NIL
4          $v.label = ClusterId$
5          $v.neighbor =$ REGION-QUERY($v, Eps$)
6          if $\|v.neighbor\| < MinPts$
7              EXPAND-CLUSTER($v, ClusterId, Eps, MinPts$)
8      elseif $v.label ==$ NOISE
9          $v.label = ClusterId$
```

Similiar with DFS traverse, DBSCAN clustering is also divided into two procedures, DBSCAN and EXPAND-CLUSTER. DBSCAN functions as a wrapper function in the same way as DFS. This procedure goes through each point in the data set S. If the current point u has been visited, the procedure skips it. Otherwise, further process continues. However, unlike calling DFS-VISIT on every unvisited u unconditionaly as DFS does, in DBSCAN, another procedure EXPAND-CLUSTER is called if and only if u has sufficient number of neighbors in a given range Eps. REGION-QUERY(u, Eps) returns all the neighbors of u with a distance no further than Eps. This is the extra condition mentioned earlier. Similiarly, this condition is also checked in EXPLAND-CLUSTER at line 6, as opposed to DFS-VISIT. Besides these two places, the rest of the two algorithms are the same. Details of extra conditions are fully explained in next section.

2.2.2 Notions of Density in Clusters

From the view of density, points in a space can be classified into three categories: core points, border points, and outliers/noise. The classification criterion on a point u bases on the size of its *Eps-neighborhood*, denoted as $N(u; Eps)$. The $N(u; Eps)$ represents the collection of all points whose distances to u are within Eps, which is specified by user. More formally, $N(u; Eps) = \{v \mid dist(u, v) < Eps\}$.

Based on above definition, a point u is a core point if and only if $\|N(u; Eps)\| \geq MinPts$, where both Eps and $MinPts$ are user specified. After defining core points, it is relatively easy to define the other two categories. A border point v is a neighbor point of a core point u, but v is not a core point itself. Formally, $\|N(u; Eps)\| < MinPts$, and $v \in N(u; Eps)$, where $\|N(u; Eps)\| \geq MinPts$. For a outlier, it's a point v which is neither a core point itself nor a neighbor point of a core point. An example graph is showing in Fig 2.3.

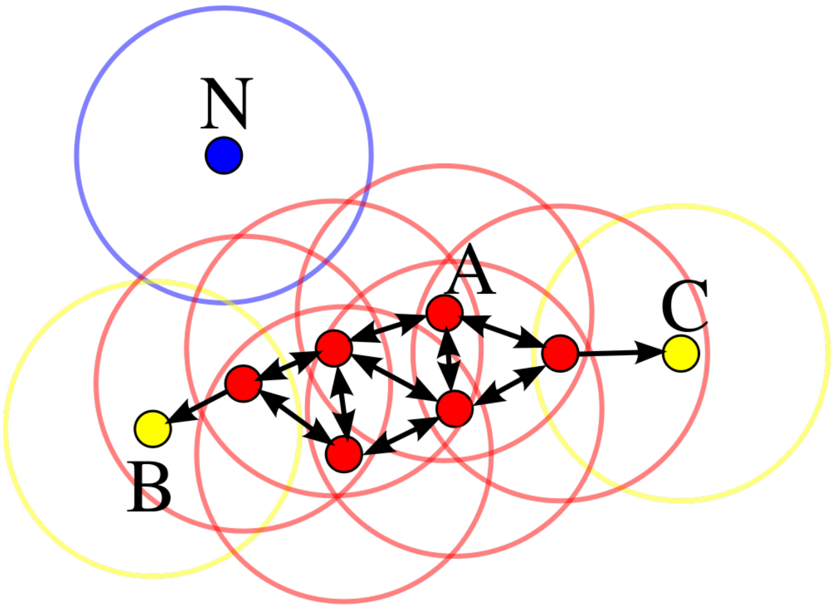

Figure 2.3: Illustration of core points(red), border points(yellow), and outlier(blue) [22]. In the illustration, $MinPts = 4$, Eps is the radius of those circles.

As show in the figure, red points are core points since in each red circle, there are at least 4 points(including the center point). While for the yellow point, they don't have enough points in their circles, but they embed in one of the red circles. Thus, they are border points. As for the blue point, it has neither enough point in the blue cirle nor is in any red circle. Thus, the blue point an outlier/noise.

These definitions reveal the assumption and theory of DBSCAN. DBSCAN assumes that, each cluster consists of two types of points, core points and border points. Core points form inner denser areas compared to spaces outside of clusters. Border points form a slightly sparser areas surrouding the inner core ares, but still the density is higher than spaces between different clusters, possesed by noise. In DBSCAN, the density of an area is measured by the number of points centered at u within an area specified by the given radius *Eps*. If there are at least $MinPts$ points, this area is considered to be a dense area. Thus, this area is part of a cluster. In the procedure EXPAND-CLUSTER, the algorithm will explore the whole dense area and less dense border area belonging to the same cluster. Each call of EXPAND-CLUSTER visits a different cluster. Thus, when the whole procedure DBSCAN finishes, different clusters are formed.

2.2.3 Analysis of DBSCAN

DBSCAN applies to a broader range of problems comapred to K-Means. In K-Means, one of the obstacle is to compute the mean value. DBSCAN doesn't have this problem as long as the distance between points is computable. This enables DBSCAN to use more compliated distance measurement, such as edit distance, which is very useful in dealing with strings. Another advantage of DBSCAN is its robustness to noises. As mentioned in section 2.1.2, the mean value will be heavily interfere by noises/outliers. In contrast, DBSCAN has a notion noises and it is able to spot these outliers, which is a core problem of this thesis. Finally, the most important advantage of DBSCAN is that it can find arbitrarily shaped clusters. It can even seperate a cluster completely surrounded by a different cluster. An example is shown in Fig 2.4.

Simliar to K-Means, DBSCAN also requires user specified parameters. To get the optimal result, a careful choice of these parameters is needed. Finding the appropriate parameter can be achieved in the similiar way by finding the elbow point as mentioned in section 2.1.2. The user can pick a number for $MinPts$ first. Then, for each point, the distance from its kth nearest neighbor is computed. After sorting these distances in descending order and plot them, a graph called *sortedk-distgraph* can be obtained. This graph reveals insights about the density distribution of the whole dataset reflected in how the *k-dist* varies. Then the *Eps* can be set to the value corresponding to the elbow point. This heuristic works well as the graph won't differ too much for $k > 4$. An illustration of this approach is shown in Figure 2.5. [7]

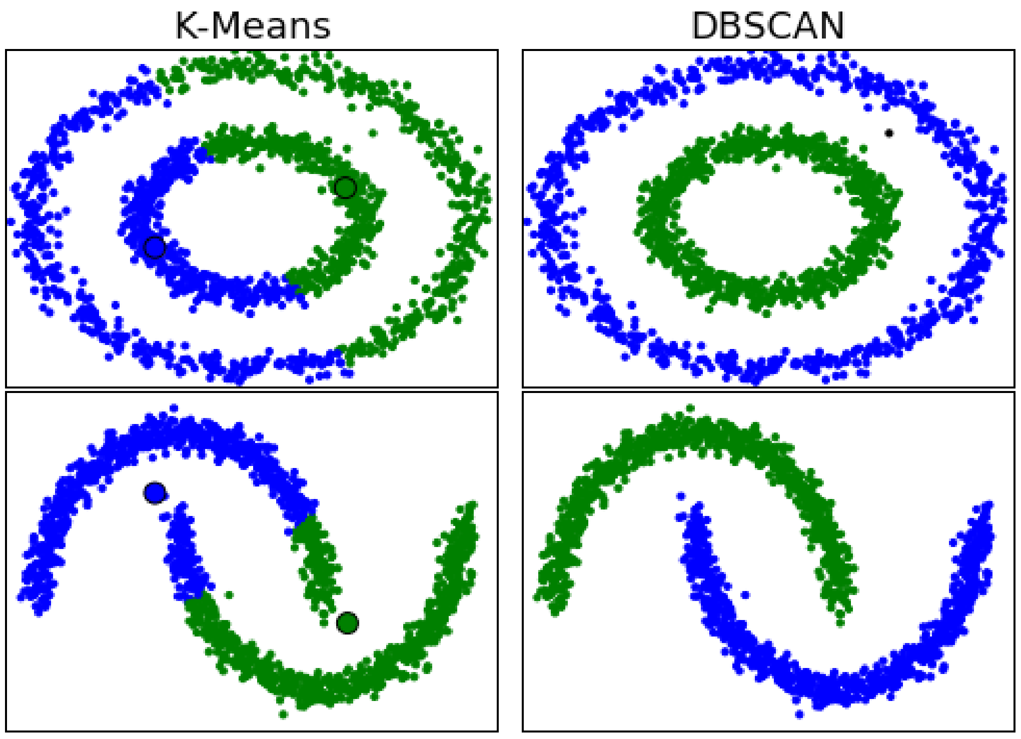

Figure 2.4: A Comparison of K-Means and DBSCAN. In the diagram, two datasets "circles" and "moons" are used. Compared to K-Means, DBSCAN gives more reasonable clustering results on these two irregular shaped datasets.

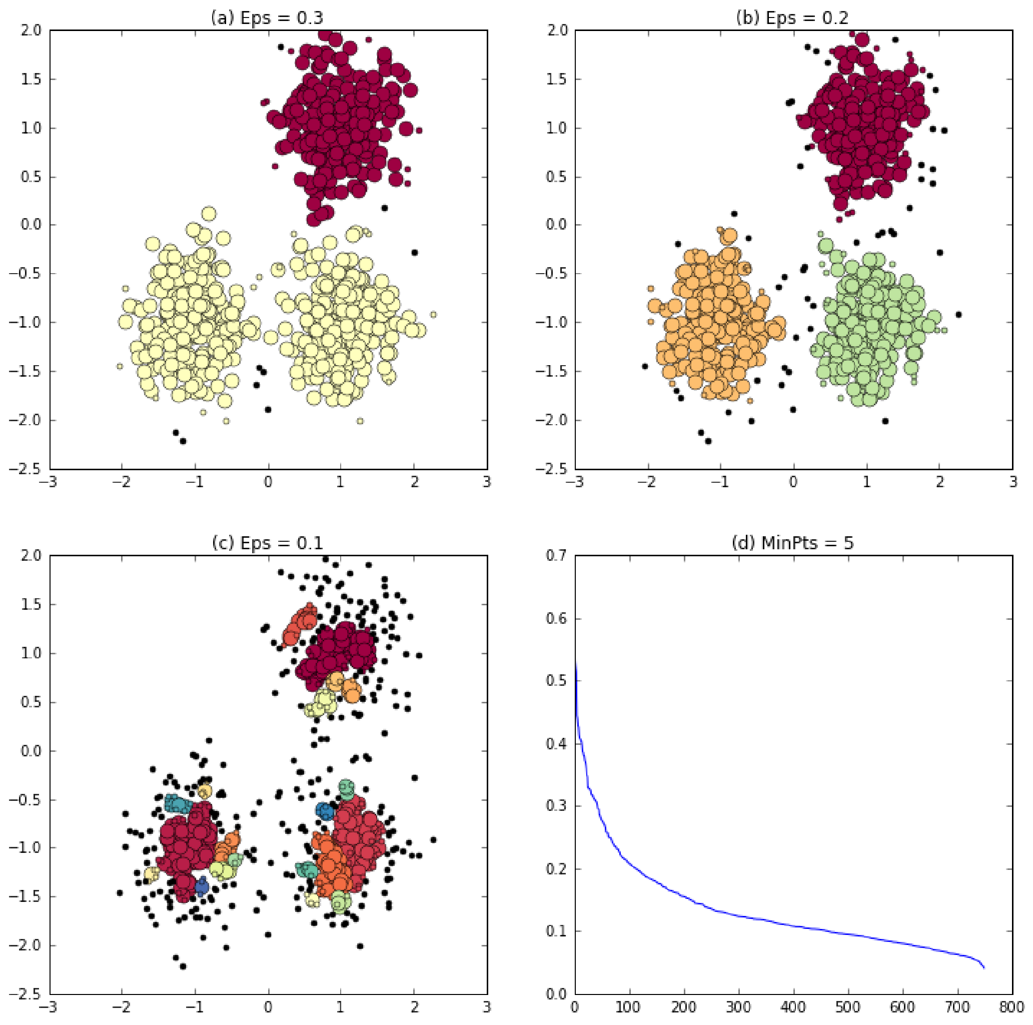

Figure 2.5: Illustration of the heuristic parameter choosing approach for DBSCAN. (a)-(c) illustrates clustering results obtained by using different *Eps*. Each cluster is painted with a different color. Core points are represented using larger circles while border points are represented using smaller circles. Black small circles represent outliers. In figure (a), very few points are labeled as noise and the bottom two clusters are not distinguished. In figure (b), the result is more reasonable and can be considered as optimal. In figure (c), too many points are labeled as noise and more than 3 clusters are reported. According to the *k-distgraph* shown in (d), 0.2 should be the best value for *Eps*, which corresponds to the result in figure (b).

Chapter 3

Scoring Patient Visits by Generative Models

This chapter is going to explore generative methods for anomaly detection. Generative methods are a collection of algorithms which try to build a model that explains the process of how the data generates. Then, based on information from the proposed model, anomalies are detected. A family of algorithms belonging to this category is the Markov models.

As described in Chapter **??**, the patients visits can be seen as time sequential data consisting of a series of events. The events in one visit are not generated independently and randomly. Instead, past events have an effect on the type of the next possible event. To handle sequential data, Markov models are the correct choice since they consider the relation between consecutive observations. In the following context, Section 3.1 introduces the basic Markov chain model. Later, Section 3.2 expands the Markov chain to a more complicated Hidden Markov Model by introducing hidden variables.

3.1 Discrete Markov Process

Consider a system having N distinct states $S_1, S_2, \cdots, X_N$. At any time, the system will be in one of these states. After a given time period t, a series obeservation $x_1, x_2, \cdots, x_t$ can be obtained. (Without loss of generality, the following discussion assumes the variables are all scalar. The assumption holds in the rest of the context unless explicitly stated otherwise) According to the prodcut rule of probability, the joint probability distribution for this sequence of observations is

$$p(x_1, x_2, \cdots, x_t) = \prod_{i=2}^{t} p(x_i \mid x_1, \cdots, x_{i-1}) \tag{3.1}$$

The conditional probability distribution of each observation x_i depends on all observations having a smaller index than it. The above relations between the observations can be represented graphically in Fig 3.1(a). The graph is fully connected, and no independence property can be obtained from it. Now assume that each observation x_i only depends on one immediate previous observation x_{i-1}. Then the joint distribution becomes

$$p(x_1, x_2, \cdots, x_t) = p(x_1) \prod_{i=2}^{t} p(x_i \mid x_{i-1}) \tag{3.2}$$

This newly obtained model is depicted in Fig 3.1(b), and is referred as *first-order Markov chain*. The term *first-order* indicates the dependence on only one previous observation. Suppose the system has only 3 states, as shown in Fig 3.1(c). Then, to fully represent the system, the only required information is the transition probabilities between different sates. The transition probabilities are usually refered as *transition matrix*, denoted as $\mathbf{A}$. Each element A_{ij} represents the probability of transfering from state s_i to state s_j. Learning the parameters of this model is very simple. Since the states are exactly the observations, A_{ij} can be simply obatained by compute the frequency of transfering to s_j starting from s_i. The number of free parameters in this model is $K(K-1)$, where K represents the number of states in the system.

Sometimes, the observations can depend on more than one observations in the past. One simple way to achieve this is creating a higer order Markov chain. By allowing each observation to depend on previous two values, a second-order Markov chain is obtained, as shown in Fig 3.2. Then the joint distribution becomes

$$p(x_1, x_2, \cdots, x_t) = p(x_1)p(x_2 \mid x_1) \prod_{i=3}^{t} p(x_i \mid x_{i-1}, x_{i-2}) \tag{3.3}$$

Compared to *first-order Markov chain*, the *second-order Markov chain* has better capability of modelling complex relations between varaibles. In fact, the higher the order is, the more flexible the model is. However, the number of parameters grows as well, which makes the model difficult to train. For a M^{th} order markov chain, there will be $K^M(K-1)$ parameters. Because the exponential growth in number of parameters, the model gardually becomes impractial as M grows.

The markov chain model can be expanded in another way, by associating an emission distribution $\mathbf{E}_k$, $k = 1, \cdots, K$, to each state in the system. Thus, two observations x_i, y_i exist at any time, where y_i is generated depending

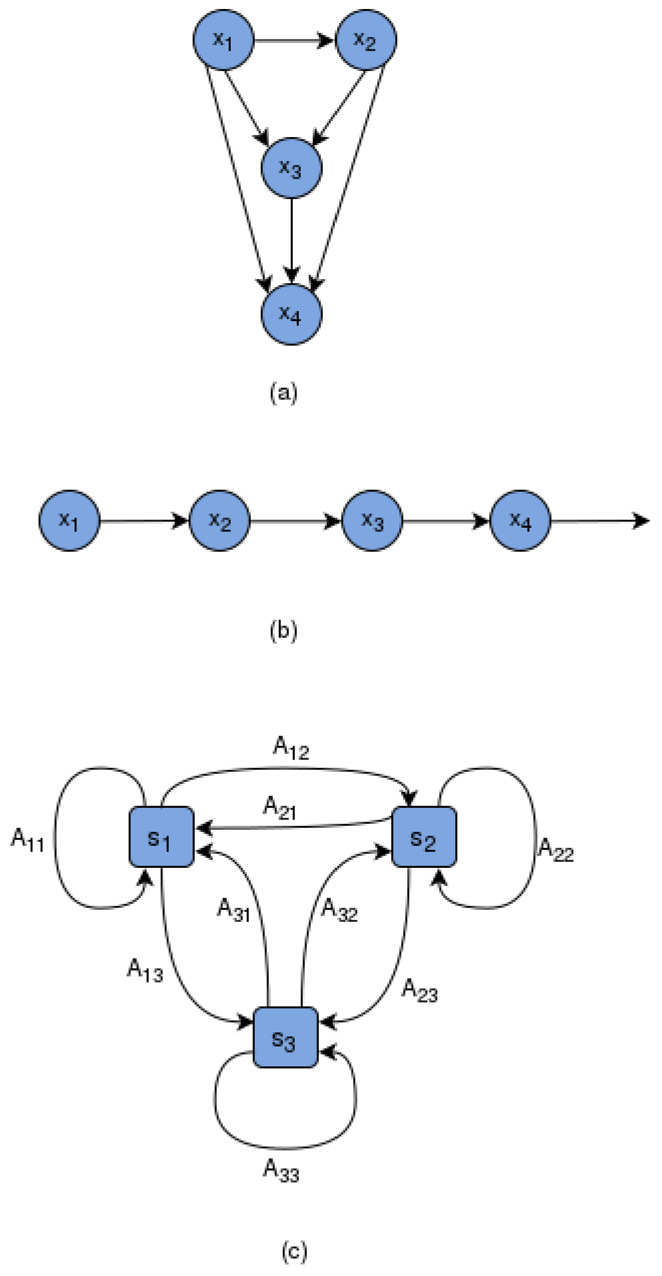

Figure 3.1: Illustration of a Markov Chain of 4 observations possessing 3 states. Variables are represented using filled circles, while states are represented using filled squares.

on x_i. If the expansion is based on a first-order Markov chain, the joint distribution becomes

$$p(x_1, y_1, \cdots, x_t, y_t) = p(x_1)\big(\prod_{i=2}^{t} p(x_i \mid x_{i-1})\big)\big(\prod_{i=1}^{t} p(y_i|x_i)\big) \tag{3.4}$$

The patient visits can be seen as an example of this expanded model. For each visit, the patient will go through a series of events, which can be considered as states. Each event will then last for a certain period of time. The duration can be seen as generated from a distribution, and the parameters of this distribution depend on the event. The example is shown in Fig 3.3.

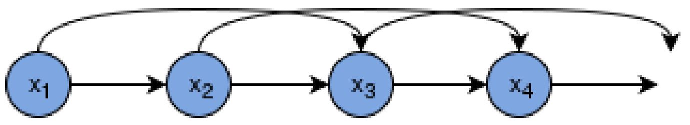

Figure 3.2: Illustration of a second-order Markov model.

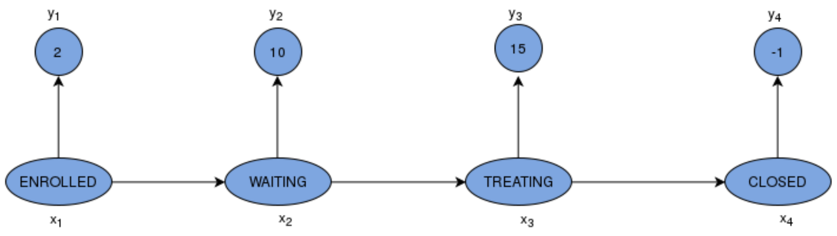

Figure 3.3: Modelling a patient visit as an expansion of first-order Markov model.

3.2 Hidden Markov Models

As mentioned in the last section, a tradeoff between flexibility and practicality exists in deciding the order number for a Markov chain model. In special situations, such as speech recognition, very high-order Markov chain is expected. But the computation resource requires as few number of parameters as possible. It would be ideal if a model is not limited to any specific given order, and still only limited number of parameters are required to specify the model. Luckily, these requirements can be satisfied by constructing a Hidden Markov Model using additional latent variables.

Suppose a sequence of observations $\mathbf{X} = \{x_1, \cdots, x_N\}$ is obtained. In stead of assuming each observation depends directly on a specific number of previous observations, the new assumption is that, there is a latent variable z_i corresponding to each observation, and the latent variables form a Markov chain. The latent variables don't have to possess any physical meanings. They can even be of different type to the observations, in terms of distribution and dimensionality. A graphical representation of this model is shown in Fig 3.4. It's easy to get confused by comparing Fig 3.3 and Fig 3.4 since they share the same graphical structure. The difference is that, in Fig 3.4, the z_i's are unobserved latent variables, which is depicted using unfilled circles. While in Fig 3.3, both events and duration are observed values. All observed variables are represented using filled circles. If the HMM is applied to the patient visits, one potential structure could be the one shown in Fig 3.5. In this model, the value of latent variable determines which event will generate, then the event determines how long the duration will be. Notice that the latent variables don't have any associated physical meaning or specific distribution form. One can explain them as indication of the functioning statu of the system by selecting them to be binary variables. When $z_i = 1$, it indicates the queue system in the hospital is working in normal mode. When $z_i = 0$, it means the system is working in a problametic way.

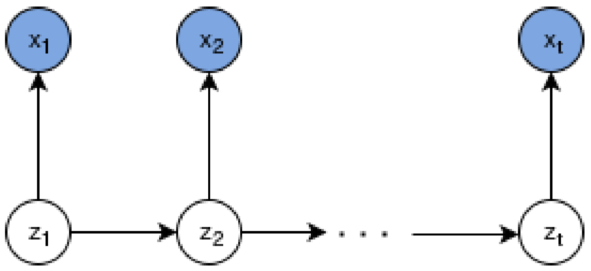

Figure 3.4: Graphical representation of a Hidden Markov Model. Observations are represented using filled circles, whilc latent variables are depicted using unfilled circles. The latent variables form a first-order Markov chain.

In the framework of HMM, the joint distribution over both observed and latent variables is given below

$$p(\mathbf{X}, \mathbf{Z} \mid \boldsymbol{\theta}) = p(z_1 \mid \pi)\Big(\prod_{n=2}^{N} p(z_n \mid z_{n-1}, \mathbf{A})\Big)\Big(\prod_{m=1}^{N} p(x_m \mid z_m, \phi)\Big) \qquad (3.5)$$

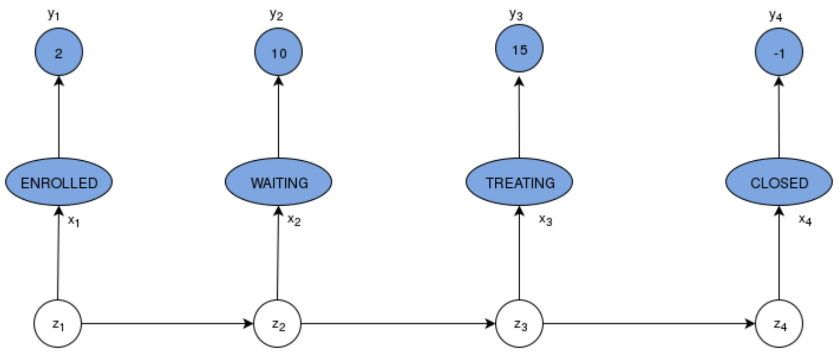

Figure 3.5: Modelling patient visit as Hidden Markov Model. Event types and event duration are represented using filled eclipses and circles respectively. Additional hidden variables are represented using unfilled circles. No specific physical meaning is associated with these latent variables.

where $\mathbf{X} = \{x_1, \cdots, x_N\}$ represents all the observed variables, $\mathbf{Z} = \{z_1, \cdots, z_N\}$ represents latent variables, and $\boldsymbol{\theta} = \{\pi, \mathbf{A}, \phi\}$ represents the parameters in this model. The π is a prior distribution for deciding the value of the first variable z_1. The matrix $\mathbf{A}$ is the transition matrix among the latent variables. The ϕ are the parameters of the emission distribution associated with z_i and x_i.

3.2.1 Learning and Inference

There are three basic problems in HMM. [21] These problems are described below using above notations:

- Problem 1: Given a sequence of observations $\mathbf{X} = \{x_1, \cdots, x_N\}$, what is the probability $p(\mathbf{X} \mid \boldsymbol{\theta})$over the observations, under specific parameters $\boldsymbol{\theta} = \{\pi, \mathbf{A}, \phi\}$?

- Problem 2: What's the value of the parameters which maximizes $p(\mathbf{X} \mid \boldsymbol{\theta})$?

- Problem 3: Given a sequence of observations $\mathbf{X} = \{x_1, \cdots, x_N\}$, what is the value of the corresponding latent variables?

The rest of the context focus on the first two questions. The reason is that, once the value of $p(\mathbf{X} \mid \boldsymbol{\theta})$ is computed, the decision on whether a given sequence is anomaly can be made by comparing $p(\mathbf{X} \mid \boldsymbol{\theta})$ to a threshould value.

Though it seems more intuitive that finding a way to evaluate $p(\mathbf{X} \mid \boldsymbol{\theta})$ should come before maximizing it with respect to the parameters, it would be more convenient to start at solving problem 2. After solving problem 2, the solution to the first problem will appear naturally. The following discussion begins by introducing some new concepts and notations.

The distribution over only observed variables $p(\mathbf{X} \mid \boldsymbol{\theta})$ is usually refered as *incomplete likelihood*, while distribution over both observed and unobserved variables $p(\mathbf{X}, \mathbf{Z} \mid \boldsymbol{\theta})$ is refered as *complete likelihood*. Using formula (3.5), the logarithm of incomplete likelihood can be represented as

$$\begin{aligned}\ln p(\mathbf{X} \mid \boldsymbol{\theta}) &= \ln \sum_{\mathbf{Z}} p(\mathbf{X}, \mathbf{Z} \mid \boldsymbol{\theta}) \\ &= \ln p(z_1 \mid \pi) + \ln \sum_{\mathbf{Z}} \Big(\big(\prod_{n=2}^{N} p(z_n \mid z_{n-1}, \mathbf{A}) \big) \big(\prod_{m=1}^{N} p(x_m \mid z_m, \phi) \big) \Big)\end{aligned} \tag{3.6}$$

The above formula is a generalization of the *mixture distribution* [3]. Maximizing this formula with respect to the parameters is very difficult since the derivatives don't have a closed form. An alternative practical working algorithm is the *expectation-maximization(EM)* algorithm [6][19]. The EM algorithm is very similiar to the K-Means algorithm mentioned in Chapter 2. The algorithm consists of two steps, E-step and M-step. In the E-step, the algorithm fixes the value of parameters and find the posterior distribution of the laten variables $p(\mathbf{Z} \mid \mathbf{X}, \boldsymbol{\theta}^{old})$. Here the notation is adopted from Bishop's [3]. The superscription *old* in θ^{old} means the parameter is fixed. Then the algorithm computes the expectation of the logarithm of the complete likelihood, with respect to the derived posterior distribution. The newly derived term becomes a function of $\boldsymbol{\theta}$, which is shown below

$$Q(\boldsymbol{\theta}, \boldsymbol{\theta}^{old}) = \sum_{\mathbf{Z}} p(\mathbf{Z} \mid \mathbf{X}, \boldsymbol{\theta}^{old}) \ln p(\mathbf{X}, \mathbf{Z} \mid \boldsymbol{\theta}) \tag{3.7}$$

Then in the M-step, the new value of $\boldsymbol{\theta}$ is updated by maximizing $Q(\boldsymbol{\theta}, \boldsymbol{\theta}^{old})$. Compared to K-Means, the E-step corresponds to assign each point to a cluster prototype, and the M-step corresponds to update the value of the prototypes. These two steps are excecuted alternatively until convergence or maximum number of iteration is reached. In the rest context, $\gamma(\mathbf{z}_n)$ and $\gamma(\mathbf{z}_{n-1}, \mathbf{z}_n)$ are introduced which stands for the posterior distribution of a single latent variable and the joint posterior distribution over two consecutive latent variables, seperately. Instead of assuming the latent variables are scalar, here they are represented using *1-of-K* coding. Namely, each latent variable is a length K vector, where one and only one of these K elements equals 1. When $z_{nk} = 1$, it means the nth latent variable is in the kth state. Using this representation schema, following equations are obtained

$$\gamma(\mathbf{z}_n) = p(\mathbf{z}_n|\mathbf{X}, \boldsymbol{\theta}^{old}) \tag{3.8}$$

$$\gamma(\mathbf{z}_{n-1}, \mathbf{z}_n) = p(\mathbf{z}_{n-1}, \mathbf{z}_n|\mathbf{X}, \boldsymbol{\theta}^{old}) \tag{3.9}$$

$$\begin{aligned} Q(\boldsymbol{\theta}, \boldsymbol{\theta}^{old}) =& \sum_{k=1}^{K} \gamma(z_{1k}) \ln \pi_k + \sum_{n=2}^{N}\sum_{j=1}^{K}\sum_{k=1}^{K} \gamma(\mathbf{z}_{n-1}, \mathbf{z}_n) \ln A_{jk} \\ &+ \sum_{n=1}^{N}\sum_{k=1}^{K} \gamma(z_{nk}) \ln p(x_n|\phi_k) \end{aligned} \tag{3.10}$$

Computation in the M-step is relatively easy. Assume the E-step has been done, so that $\gamma(\mathbf{z}_n)$ and $\gamma(\mathbf{z}_{n-1}, \mathbf{z}_n)$ are like constants now. Then following update equation can be obtained

$$\pi_k = \frac{\gamma(z_{1k})}{\sum_{j=1}^{K} \gamma(z_{1j})} \tag{3.11}$$

$$A_{jk} = \frac{\sum_{n=2}^{N} \gamma(z_{n-1,j}, z_{nk})}{\sum_{l=1}^{K}\sum_{n=2}^{N} \gamma(z_{n-1,j}, z_{nl})} \tag{3.12}$$

Updation of ϕ_k is more tricky, since it depends on the specific choice of the emission distribution. One good observation is that, only the final term depends on ϕ_k, and different ϕ_k doesn't couple with each other. Thus, each ϕ_k can be updated seperately. The term $\gamma(z_{nk})$ functions as a soft assignment, representing the probability of assigning a point x_n to each state.

Computation in E-step is more difficult which requires efficient algorithm. The most widely used algorithm is known as *alpha-beta* algorithm. This algorithm can be seen as an application of dynamic programming technique which takes advantage of the tree structure in HMM thus leading to efficiency. To start with the alpha-beta algorithm, following conditional independence properties should be obtained first [14]

$$\begin{aligned} p(\mathbf{X}|\mathbf{z}_n) =& p(x_1, \cdots, x_n|\mathbf{z}_n) \\ & p(x_{n+1}, \cdots, x_N|\mathbf{z}_n) \end{aligned} \tag{3.13}$$

$$p(x_1, \cdots, x_{n-1}|x_n, \mathbf{z}_n) = p(x_1, \cdots, x_{n-1}|\mathbf{z}_n) \tag{3.14}$$

$$p(x_1, \cdots, x_{n-1}|z_{n-1}, \mathbf{z}_n) = p(x_1, \cdots, x_{n-1}|\mathbf{z}_{n-1}) \tag{3.15}$$

These equations can be obtained by using *d-seperation* technique [20], or proved formally using sum and product rules of probability. Using the first independence property and Bayes' theorem, following equations are obtained

$$\begin{aligned}\gamma(\mathbf{z}_n) = p(\mathbf{z}_n|\mathbf{X}) &= \frac{p(\mathbf{X}|\mathbf{z}_n)p(\mathbf{z}_n)}{p(\mathbf{X})} \\ &= \frac{p(x_1, \cdots, x_n, \mathbf{z}_n)p(x_{n+1}, \cdots, x_N|\mathbf{z}_n)}{p(\mathbf{X})} \\ &= \frac{\alpha(\mathbf{z}_n)\beta(\mathbf{z}_n)}{p(\mathbf{X})}\end{aligned} \tag{3.16}$$

where

$$\alpha(\mathbf{z}_n) = p(x_1, \cdots, x_n, \mathbf{z}_n) \tag{3.17}$$
$$\beta(\mathbf{z}_n) = p(x_{n+1}, \cdots, x_N|\mathbf{z}_n) \tag{3.18}$$

Using the other two conditional independence properties, $\alpha(\mathbf{z}_n)$ can be expressed recursively in terms of $\alpha(\mathbf{z}_{n-1})$

$$\begin{aligned}\alpha(\mathbf{z}_n) &= p(x_n|\mathbf{z}_n) \sum_{\mathbf{z}_{n-1}} p(x_1, \cdots, x_{n-1}, \mathbf{z}_{n-1})p(\mathbf{z}_n|\mathbf{z}_{n-1}) \\ &= p(x_n|\mathbf{z}_n) \sum_{\mathbf{z}_{n-1}} \alpha(\mathbf{z}_{n-1})p(\mathbf{z}_n|\mathbf{z}_{n-1})\end{aligned} \tag{3.19}$$

Similiarly, $\beta(\mathbf{z}_n)$ can also be expressed recursively as

$$\beta(\mathbf{z}_n) = \sum_{\mathbf{z}_{n+1}} \beta(\mathbf{z}_{n+1})p(x_{n+1}|\mathbf{z}_{n+1})p(\mathbf{z}_{n+1}|\mathbf{z}_n) \tag{3.20}$$

The term $\alpha(\mathbf{z}_n)$ can be seen as messages propagated from the beginning to the end. Each $\alpha(\mathbf{z}_n)$ receives messages passed from its predecessor, combines these information with its own information and then pass them to its successor. The logical also applies to the term $\beta(\mathbf{z}_n)$, but the messages are from the end to the beginning. Due to the tree strucutre in HMM, computing each term only depends on one adjacent term, instead of all terms before/after it. Thus, the computation reduces dramatically which makes the algorithm efficient. To start the whole computation, initial conditions $\alpha(\mathbf{z}_1)$ and $\beta(\mathbf{z}_n)$ are required. The initial conditions are given below

$$\alpha(\mathbf{z}_1) = \prod_{k=1}^{K} \{\pi_k p(x_1|\phi_k)\}^{z_{1k}} \tag{3.21}$$
$$\beta(\mathbf{z}_N) = 1 \tag{3.22}$$

Having obtained $\alpha(\mathbf{z}_n)$ and $\beta(\mathbf{z}_n)$, the posterior distribution $\gamma(\mathbf{z}_n)$ can be computed as in equation (3.16). As for $\gamma(\mathbf{z}_{n-1}, \mathbf{z}_n)$, it can be computed as following

$$\begin{aligned}
\gamma(\mathbf{z}_{n-1}, \mathbf{z}_n) &= p(\mathbf{z}_{n-1}, \mathbf{z}_n|\mathbf{X}) \\
&= \frac{p(\mathbf{X}|\mathbf{z}_{n-1}, \mathbf{z}_n)p(\mathbf{z}_{n-1}, \mathbf{z}_n)}{p(\mathbf{X})} \\
&= \frac{p(x_1, \cdots, x_{n-1}|\mathbf{z}_{n-1})p(x_n|\mathbf{z}_n)p(x_{n+1}, \cdots, x_N|\mathbf{z}_n)p(\mathbf{z}_n|\mathbf{z}_{n-1})p(\mathbf{z}_{n-1})}{p(\mathbf{X})} \\
&= \frac{\alpha(\mathbf{z}_{n-1}p(x_n|\mathbf{z}_n)p(\mathbf{z}_n|\mathbf{z}_{n-1})\beta(\mathbf{z}_n)}{p(\mathbf{X})} \qquad (3.23)
\end{aligned}$$

Up till now, both steps in EM algorithm are introduced, and the problem 2 can be solved efficiently. The left question is how to solve problem 1, computing the likelihood over the incomplete data. The solution comes from equation (3.16). Notice that $\gamma(\mathbf{z}_n)$ is a posterior distribution. Integrating both sides of equation (3.16) over $\mathbf{z}_n$ gives

$$p(\mathbf{X}) = \sum_{\mathbf{z}_n} \alpha(\mathbf{z}_n)\beta(\mathbf{z}_n) \qquad (3.24)$$

where $\mathbf{z}_n$ is an arbitrary latent variable. If $n = N$, then $\beta(\mathbf{z}_n) = 1$, which makes the above equation simpler

$$p(\mathbf{X}) = \sum_{\mathbf{z}_N} \alpha(\mathbf{z}_N) \qquad (3.25)$$

Then, both problem 1 and problem 2 are solved.

Chapter 4

Experiments

This chapter describes the detailed experimental implementation. Following topics will be discussed:

- Which specific data are used? How to represent the data? What preprocess has been done? What related functions are defined?
- Which methods are used? What's the reason to use such methods?

The three topics will be discussed in three sections, respectively. Analysis of the result will be postponed to Chapter ??.

4.1 Data Details and Representation

Since the akseli system has gone through several updation after its first release, many features have changed, including how the visit log is reocrded. Considering this aspect, only data generated after year 2014 is used in the experiment. All the data are from Oulu Hospital, with patient privacy information eliminated. In total, 243K unique visits with 1.93 million entries are retrieved, which spans from January to July. The retrieved data consists of three columns: visit id, event type, and recorded time of the event. Other information, such as which resource generated the data, is not retrieved. Among the retrieved data, 7 event types are adopted. The used event types are: `ENROLLING`, `WAITING`, `IN_TREATMENT_ROOM`, `PAUSED`, `IN_TREATMENT_ROOM_FROM_PAUSED`, `CLOSED`, and `CANCELLED`.

Huang et al [12] proposed to represent the patient visits as a sequence of pairs. Each pair contains the information of (a). what the event is and (b). when this event happened. For example, given a sequence showing below

$$\langle (a, 1), (b, 2), (c, 5), (d, 7) \rangle$$

it means a patient comes to the hospital, and at time 1, the patient encounters event a. Then at time 2, the patient encounters event b, and so on. The time unit can be selected arbitrarily, such as minutes, hours, or days. Huang et al used days as the time unit. In their work, they tried to cluster the patient traces. However, the patient trace has different length. Thus, typical distance metrics such as euclidean distance is not applicable. To address this problem, they also proposed a new distance metric, based on edit distance.

Edit distance is commonly used in comparing strings and biological sequences, such as proteins. Edit distance is defined as the minimum number of allowd operations used, to transform a string s to another string t. For example, if the allowed operations are *delete*, *insert*, and two strings $S =$ *"array"*, $T =$ *"xray"* are given. Then the edit distance between s and t is 3, by taking 3 operations. One potential transformation is:

1. Delete the second letter r in S by x. Then S becomes *"aray"*.
2. Delete the first letter a in S. Then S becomes *"ray"*.
3. Insert letter x at the beginning of S. Then S becomes *"xray"*, which is the same with string T.

The edit distance problem can be solved effectively by using the dynamic programming technique. Using terminology from dynamic programming, the optimal solution to the edit distance problem can be represented recursively

$$D(i,j) = \begin{cases} D(i-1,j-1) & \text{if } S[\text{i}] = T[\text{j}] \\ \min\{D(i-1,j), D(i,j-1)\} + 1 & \text{if } S[\text{i}] \neq T[\text{j}] \end{cases}$$

The edit distance only considers the difference between types of events, when applied to the patience visit data. However, the time associated with each event should also makes an effect when comparing two traces. Huang et al addressed this problem by providing a modified edit distance. In the old edit distance, events from two patient trace will either increase the distance by 1 if they belong to different type or 0 if they belong to the same type. In the modified distance, however, the increament caused by two events range from $[0,1]$ as shown below

$$\delta(\sigma_i, \rho_j) = \begin{cases} 1 & \text{if } \sigma_i(e) \neq \rho_j(e) \\ \frac{|\sigma_i(t) - \rho_j(t)|}{max\{\sigma_i(t),\ \rho_j(t)\}} & \text{if } \sigma_i(e) = \rho_j(e) \end{cases}$$

where σ and ρ are two patient traces. σ_i is the ith pair of the trace. $\sigma_i(e)$ and $\sigma_i(t)$ represent the event type and timestamp of the ith pair in that trace. The intuition of the above formula is that, if the event types of two pairs in two traces are different, then they contribute 1 to the edit distance. If the event types are the same, then the distance is determined by the timestamp associated with the two events. The closer the timestamps are, the smaller the distance is.

The modified edit distance seems reasonable. However, some subtle issues exist when the modified edit distance applies to Huang's representation. Consider two patient traces

$$S = \langle (a, 1), (b, 1000), (c, 1001), (d, 1002) \rangle$$
$$T = \langle (a, 1), (b, 2), (c, 3), (d, 4) \rangle$$

The two traces are very similiar, except that the second pair differs greatly. But this difference propagates further to the third and fourth pairs, incurring more penalty. The modified edit distance will equal almost 3. It would be more reasonable if the distance accounts only the huge different generated in the second pairs, and considers the third and fourth pair the same. To address this problem, in the experiment, a modification on the representation form is applied. Rather than record the absolute timestamp associated with each event, the duration of each event is recorded. Thus, the above two patient traces becomes

$$S = \langle (a, 1), (b, 999), (c, 1), (d, 1) \rangle$$
$$T = \langle (a, 1), (b, 1), (c, 1), (d, 1) \rangle$$

Applying the modified edit distance to the new representation, the answer equals roughly 1, which is more intuitive. Thus, in all experiments, the second representation form is adopted. Minute is used as the time unit.

Another critical point is about preprocessing. After using the second representation form, numerous noise points are observed. It's believed that the noise points are generated by the system itself for logging reasons. The feature of the noise is that all events have a 0 duration time. The noise points consist of approximately 20% of all data, which incurs great effect in training models. Thus, in the preprocess step, all noise data are manually removed.

4.2 Methods

Chapter 2 and chapter 3 introduced 4 potential methods. However, only DBSCAN and Markov Chain are used in the experiment. This section explains the reasons for choosing only these two methods and describe related details.

4.2.1 Choice of Clustering Method

As stated in Chapter 2, both DBSCAN and K-Means need a elaborate distance metric. This requirement can be fullfilled by using the modified edit distance. Besides this, K-Means also requires efficient computation of the mean value. However, based on the current data representation, it is not clear how to compute the mean. Thus, K-Means is not applicable in this setup.

One critical phase in applying DBSCAN is how to efficiently compute the *Eps-neighborhood*. The DBSCAN algorithm will go through several iterations. Each iteration involves computing *Eps-neighborhood* for all points. The efficiency of computing *Eps-neighborhood* directly determines the practicability of DBSCAN. A naive implementation is to compute a N by N pair-wise distance table. Then sort the rows in the matrix. After this, finding the *Eps-neighborhood* takes only constant time. This process takes $O(N^2)$ space and time. The space complecity can be further reduced to $O(N)$.

A more efficient method, vantage-point tree [23] can be used in the experiment. Vantage-point tree is a recursively built balanced binary tree. Each non-leaf node consists of two fields, a point functions as a center and a radius, and has two children. The center point is randomly selected from a set of available points. Initially, the set is the whole data points. Then, distances from the center to all the rest points in the set is computed. Next, the radius is set to the median of the distances. After this, the point set is divided into two subsets, one consists of points with a distance shorter than the radius, the other consists of points with a distance longer than the median value. The first set is stored in the left child of current node, and the second set is stored in the right child of the current node. Intuitively, this works as drawing a circle centered on the selected center point. The circle partitions the other points into two parts, with half inside the circle, half outside the circle. This process continues until a leaf node is encountered. This building process takes $O(N \log N)$ time and $O(N \log N)$ space.

After building the vantage-point tree, finding the *Eps-neighborhood* for a single point takes $O(\log N)$ time. Suppose the query point is p. Now the query enters a node centered on point q and the radius of the node is τ. Suppose the distance from p to q is $\delta < \tau$. Then the query explores only the left child of q if $\delta + \epsilon \leq \tau$. Otherwise, the right child is also explored. The intuition is that, if one *Eps-neighborhood* of query point p is exactly ϵ far to p, then the distance between this neighbour point and the centerd point q is at most $\delta + \epsilon$. If this distance is no larger than the radius associated to the centered node, then there is no need to explore points outside the circle, which reduces the search space by half. As a result, find *Eps-neighborhood* for all points takes $O(N \log N)$ time.

4.2.2 Choice of Generative Method

In the experiment, the simple Markov Chain model is selected for several reasons. Firstly, the patient trace is not long. A typical patient trace has only four events, `ENROLLING`, `WAITING`, `IN_TREATMENT_ROOM`, and `CLOSED`. The longest patient trace doesn't have more than 10 events. Even building an explicit high-order Markov Chain model on these data will still be practical. Another point is that, the type of potential event closely depends on the previous one, rather than depending on an implicit statu. For example, after the event `ENROLLING`, it is very likely the next event is `WAITING`. Sometimes, due to special situations such as cancellation of the reservation, the event `CLOSED` follow. But it is impossible a `IN_TREATMENT_ROOM` event comes, which skips the `WAITING` phase. The last argument is that, anomaly happened in one phase does not necessarily affect the coming phase. For example, a patient may encounter problems while being in the `WAITING` phase. But this does not mean the patient may also encounter problems while in `IN_TREATMENT_ROOM` phase. In other words, being in abnormal status in each phase is independent, which is quite contrary to the assumption of HMM that the value of current hidden variable depends on value of the previous hidden variables. As a result, a simple one-order Markov Chain model is selected in the experiment.

In the one-order Markov Chain model, the events are represented using *1-of-K* coding schema. Thus, the events are multinomial variables. Another important part of the model is the choice for the emission distribution. Usually, Gaussian distribution is used for modelling continuous variables. However, waiting time is not symmetrically distributed with respect to a mean. The shortest waiting time can only be 0 and the longest waiting time can be very long. The distribution has a skew with a long tail. According to studies in queuing theory, Poisson distribution is more appropriate. Several methods exist to test if Poisson distribution applies, for example, measuring the dispersion. Before stepping into numerical computations, visualizing the distribution is a good start. Related histogram of duration distribution associated with each event is shown in Fig 4.1. Intuitively, the distributions of `ENROLLING`, `IN_TREATMENT_ROOM`, and `IN_TREATMENT_ROOM_FROM_PAUSED` look like Poisson distribution. The distribution of `WAITING` seems rather different, whcih resembles mixtrue of two Poisson distributions. One plausible explanation is that, these data is obtained from all departments in the hospital. It is possible different departments have different typical waiting time. Luckily, information of department can be also obtained. After extracting data from the largest department, 8915 entries remains. The histogram of these data is shown in Fig 4.2. The newly obtained data seems to be more likely from Poisson distribution. Notice that, in these data, only three events remains.

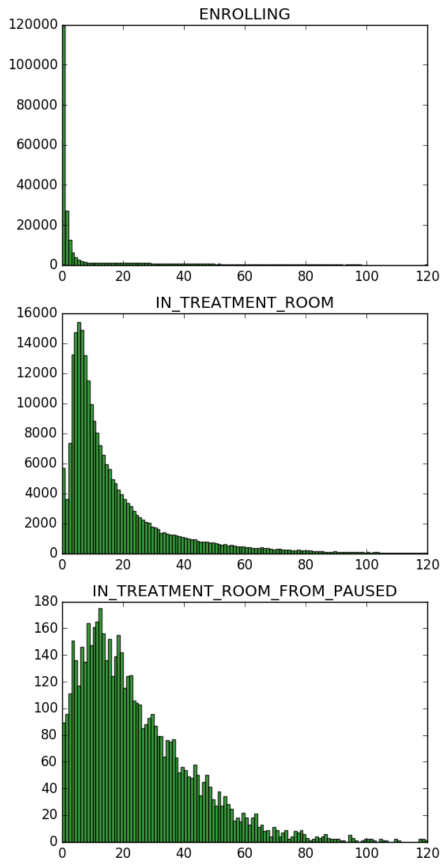

Figure 4.1: Histogram of duration associated with all events

Next is to compute the dispersion to see if the data distribution really matches Poisson distribution. The formula to compute dispersion is as follow

$$D = \frac{\sigma^2}{\mu}$$

where σ^2 and μ is the variance and mean value of the data, respectively. One caveat is very large duration time in each distribution. For example,

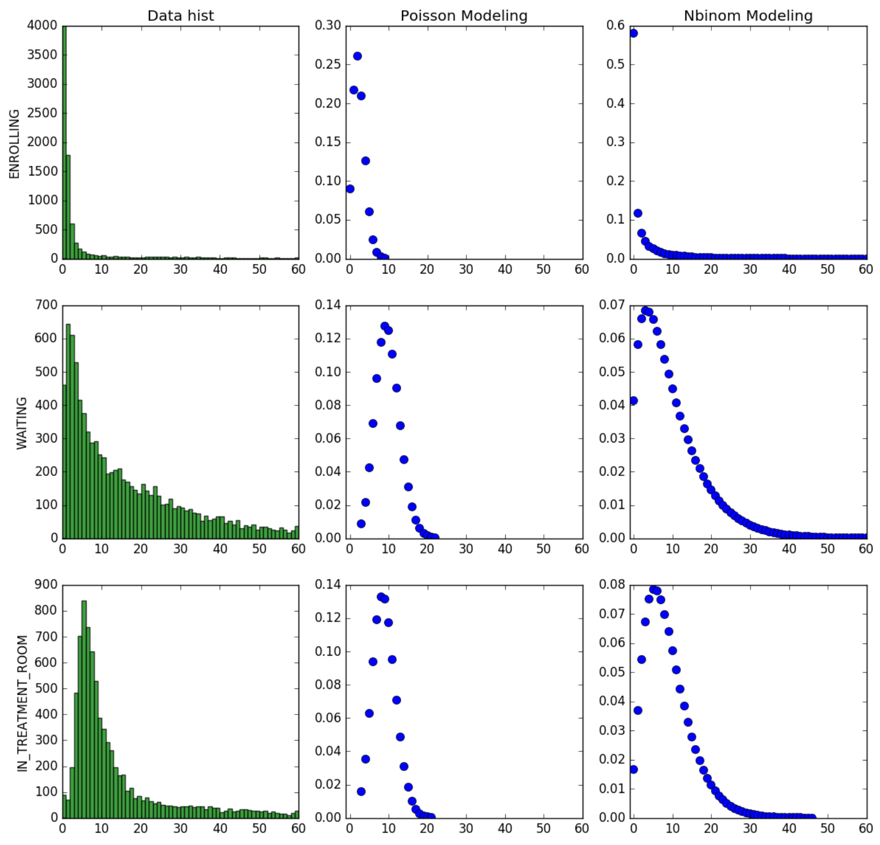

Figure 4.2: Histogram of duration associated with all events using data generated by the largest department, together with Poisson distribution fitting and negative binomial distribution fitting.

the longest duration in `WAITING` can be over 1000. Though this situation is very rare, it incurs very large difference when computing μ and σ. In fact, such rare values can be considered as the anomalies we are trying to detect. Thus, it makes sense to ignore these cases. It's hard to decide a threshould that determines which part of data should be discarded. In the experiment, the threshould is set to 30. All duration longer than this threshould is not used while computing μ and σ. The reason of selecting 30 is that a typical event seldomly lasts more than 30 minutes. In fact, 84% events in this data have less than 30 minutes duration. The computed values are listed in table 4.1 When dispersion equals 1.0, it means the distribution follows Poisson

events	mean	variance	dispersion
`ENROLLING`	2.4	28.9	12.0
`WAITING`	9.8	68.0	6.93
`IN_TREATMENT_ROOM`	8.9	36.0	4.0

Table 4.1: Mean, variance, and dispersion of data generated from largest department.

distribution. Typically, data generated from Poisson distribution can have slightly larger dispersion than 1.0. However, as shown in table 4.1, the dispersions are much larger than 1.0. This suggests that the distributions are more likely to come from negative binomial distribution rather than Poisson distribution. Compared to Poisson distribution, negative binomial distribution has "heavier" tails and larger variance. Considering this aspect, negative binomial distribution modelling is also implemented in the experiment. The fitting result is shown in Fig 4.2. As shown in the figure, negative binomial distribution has a better fitting of the data. Thus, in the final, first-order Markov Chain with negative binomial distribution is chosen as the generative method.

Chapter 5

Results and Discussion

Since the data doesn't have labels indicating which entries are anormalies, numerical evaluation is not possible. To examine how the methods perform, user interpretation of the data is the only standard. To help strength intuitive understanding, a visualization method t-SNE[17] was applied. This method only requires a distance metric between pairs of data entries. Then it projects the entries into a 2D space showing potential structures underlying the data. t-SNE typically places "important" points in the center of the drawing. The concept of "importance" can be understand as a special kind of density. Intuitively, points lying in dense area are less likely to be outliers.

Section 5.1 and 5.2 describes results obtained using clustering method and generative method respectively. Then, section 5.3 compares these two methods.

5.1 Clustering Method Results

The first step of applying DBSCAN is to choose parameters. As stated in section 2.2.3, DBSCAN is relatively robust with $k > 4$, where k stands for the kth nearest neighbor. In this experiment, k was set to 7. The distance of the 7th nearest neghbor of all points is draw in Fig 5.1. The figure shows that the distance begins to increase dramatically beyond 0.5. Thus, in the experiment, Eps was finally set to 0.5, with $MinPts = 7$. The clustering result is shown in Table 5.1 and visualized in Fig 5.2.

As shown in the table, 12 clusters are generated by DBSCAN. Among the 12 clusters, cluster 0 is labeled as noise/anomalies which consists of 352 visits. Cluster No. 2 is the largest cluster which consists of 7547 visits. This cluster is believed to consists of the most typical visits. For the rest small clusters, they can be considered as representing some non-typical but normal

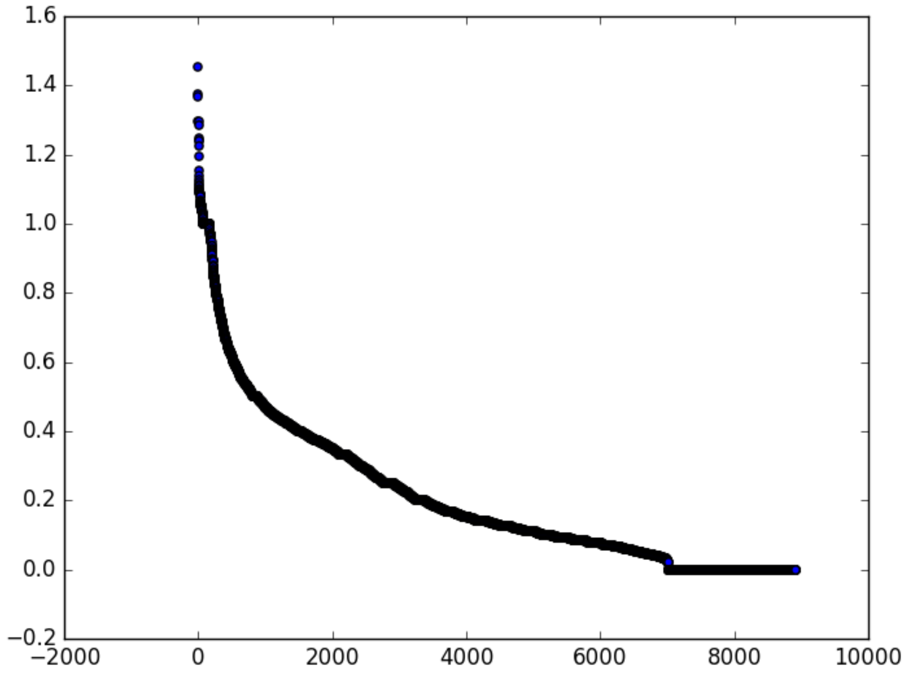

Figure 5.1: 7th nearest neighbor distance for selecting *Eps* of DBSCAN

Cluster No.	0	1	2	3	4	5	6	7	8	9	10	11
Num Points	352	10	7547	134	215	283	64	104	44	47	21	94

Table 5.1: Mean, variance, and dispersion of data generated from largest department.

visits. One potential reason of generating such subclusters is that, there are many different resources/machines for diagnosing. Data in these subclusters are generated from these less frequently used resources/machines.

In Fig 5.2, cluster 0 is plotted using red while the rest clusters are all painted in blue. As the figure shows, many red points located on the border of the figure, which indicates they come from a sparser area in their original space and have less importance. This corresponds the intuition interpretation.

To further verify the suspection, 10 samples from each cluster are listed in Table A.1. Visits in Cluster 0 seem very uncommon. Some visits just ended

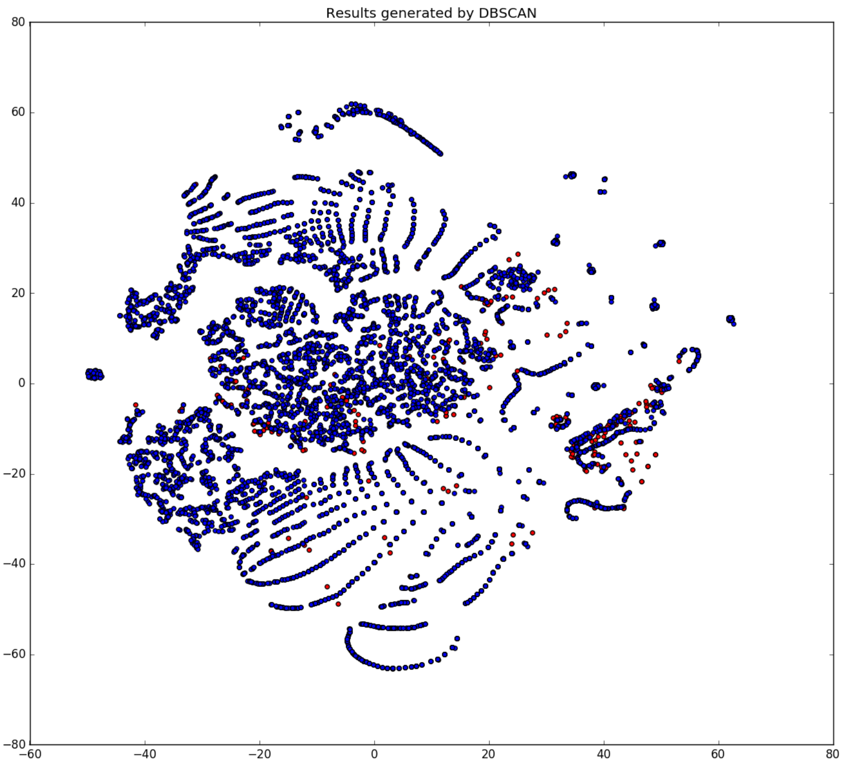

Figure 5.2: Results generated by DBSCAN, visualized using t-SNE

without neither closed by the doctor nor cancelled by the patient. Cluster 1 consists of similiar visits. Cluster 2 seems to have many reasonable visits. Visits from this cluster typically consists of four events and each event has duration no longer than 30 minutes. Thus, this cluster can be interpreted as the collection of normal visits as assumed. The rest clusters also exhibit some intuitive patterns. Some small clusters can be also considered as anomalies in addition to cluster 0, for example, cluster 10. The reason there are many clusters is that the distance between border points in two clusters are too large so that the two clusters did not merge.

5.2 Generative Method Results

Compared to DBSCAN, Markov Chain method is easier to implment. However, some special rules need to be set before running the method. The first rule is how to handle the -1 appeared in each event, which means the visit terminates. Since negative binomial only defines probability for non-negative inputs, the probability of duration equals -1 is undefined. In the experiment, we set this probability to be the frequency of being -1 happened in this event. So the summation of the probability of all possible values is slightly larger than 1. But this does not incur large effect.

Another caveat is the numerical issues while computing likelihood. Since the likelihood of a probability will typically be so small that precision problem may occur. To avoid this, the log-likelihood is computed instead. Visits having longer sequence of events tend to have smaller likelihood, but this does not mean the visit is less likely to happen. Considering this problem, the final log-likelihood is normalized by dividing the length of the sequence. The log-likelihood of visits sorted in decreasing order is shown in Fig 5.3. As shown by the figure, most visits have a log-likelihood larger than -10, the rest few visits with log-likelihood much smaller than -10 are very like to be anomalies. After selecting a threshould to be -10, the detection result is shown in Fig 5.4. Anomalies are painted in red. Again, these suspected anomalies locates on border areas in the figure.

Similiarly, for every 1000 visits, 5 samples are selected with their log-likelihood listed in Table A.2 for further exploration. As listed in the table, visits in the first several blocks are very similiar and seem to be very normal. It was only from the last but one block, visits start to behave in different ways. And in the last block, which consists of visits have very large negative values of log-likelihood, these visits are very bizzare and are exactly the anomalies we tried to find.

5.3 Discussion

Above result suggests both DBSCAN and Markov Chain can spot out anomalies. However, compared to DBSCAN, we think Markov Chain is a better method for several reasons.

Firstly, clusters formed by DBSCAN are sligtly contaminated. For example, the 4th visit in cluster 7 seems very abnormal and should appear in other clusters. Other clusters also have entries does not resemble other visits in this log. A reason for such behavior is the hard assignment to clusters in DBSCAN. In Markov Chain, however, each visit is assigned by a score which

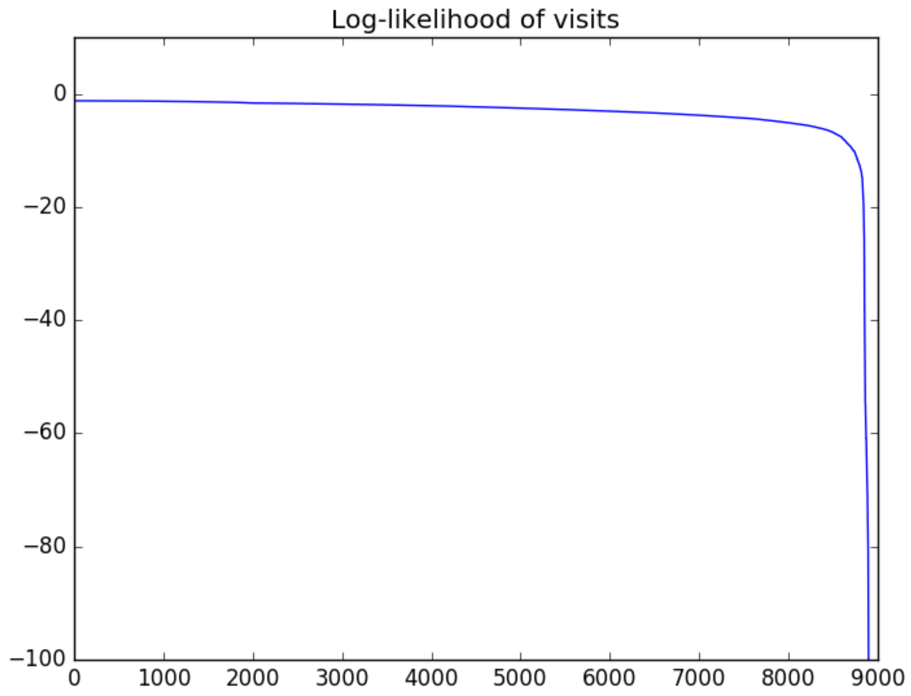

Figure 5.3: Log-likelihood of all visits sorted in decreasing order.

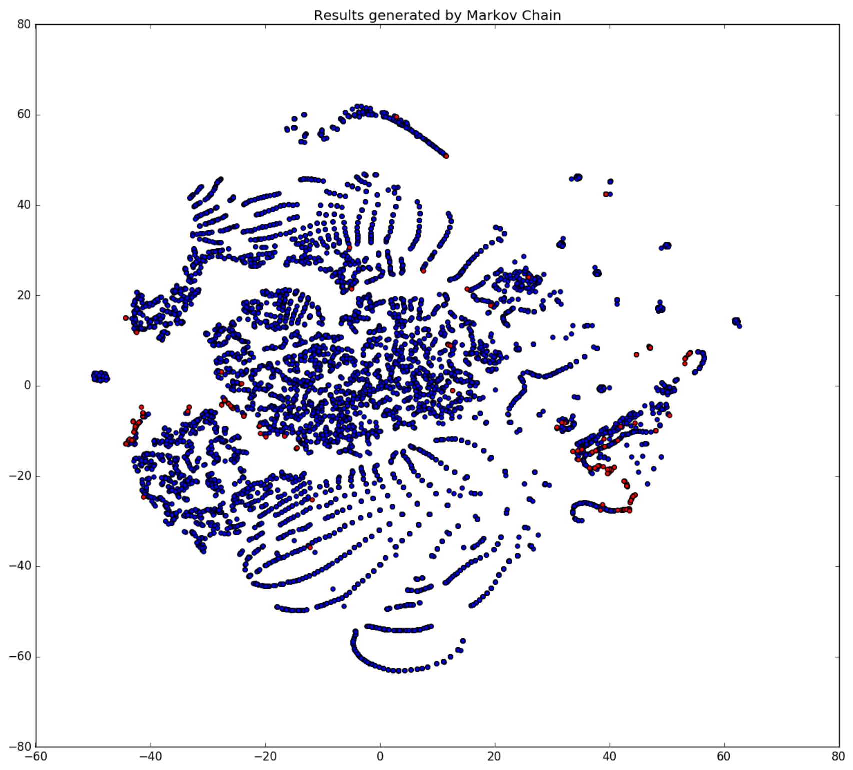

Figure 5.4: Results generated by first-order Markov Chain using negative binomial distribution as emission function, visualized using t-SNE. Anomalies are painted in red.

indicates how “normally” this visit is. This “soft-assignment” is a better description of the entries. Besides, the user has to interpret the meaning of each cluster by themselves, which is typically unexpected by the user.

Secondly, Markov Chain has better time and space complexity for detecting future anomalies. When determine if a new visit log is anomaly, DBSCAN will compare this new visit to all past visits and then assign this new visit to the cluster fitting it best. Thus, DBSCAN requires to maintain all past visits, and new detection takes $O(n)$ time. This requirement will gradually becomes impractical. In contrast, Markov Chain only needs to maintain the computed parameters of transition matrix and emission functions. Computing the log-likelihood takes $O(1)$ time for each new visit log. Thus, speaking from this aspect, Markov Chain is a much better method than DBSCAN.

Bibliography

[1] Aggarwal, C. C. An introduction to outlier analysis. In *Outlier Analysis*. Springer, 2013, pp. 1–40.

[2] Aloise, D., Deshpande, A., Hansen, P., and Popat, P. Np-hardness of euclidean sum-of-squares clustering. *Machine learning 75*, 2 (2009), 245–248.

[3] Bishop, C. M. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

[4] Campello, R. J., Moulavi, D., Zimek, A., and Sander, J. Hierarchical density estimates for data clustering, visualization, and outlier detection. *ACM Transactions on Knowledge Discovery from Data (TKDD) 10*, 1 (2015), 5.

[5] Davis, J. C., and Sampson, R. J. *Statistics and data analysis in geology*, vol. 646. Wiley New York et al., 1986.

[6] Dempster, A. P., Laird, N. M., and Rubin, D. B. Maximum likelihood from incomplete data via the em algorithm. *Journal of the royal statistical society. Series B (methodological)* (1977), 1–38.

[7] Ester, M., Kriegel, H.-P., Sander, J., Xu, X., et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Kdd* (1996), vol. 96, pp. 226–231.

[8] Forsyth, D. A., and Ponce, J. *Computer vision: a modern approach.* Prentice Hall Professional Technical Reference, 2002.

[9] Gupta, D., and Denton, B. Appointment scheduling in health care: Challenges and opportunities. *IIE transactions 40*, 9 (2008), 800–819.

[10] Gupta, M., Gao, J., Aggarwal, C., and Han, J. Outlier detection for temporal data. *Synthesis Lectures on Data Mining and Knowledge Discovery 5*, 1 (2014), 1–129.

[11] He, Z., Xu, X., and Deng, S. Discovering cluster-based local outliers. *Pattern Recognition Letters 24*, 9 (2003), 1641–1650.

[12] Huang, Z., Lu, X., and Duan, H. Anomaly detection in clinical processes. In *AMIA Annual Symposium Proceedings* (2012), vol. 2012, American Medical Informatics Association, p. 370.

[13] Hulshof, P. J., Kortbeek, N., Boucherie, R. J., Hans, E. W., and Bakker, P. J. Taxonomic classification of planning decisions in health care: a structured review of the state of the art in or/ms. *Health systems 1*, 2 (2012), 129–175.

[14] Jordan, M. I. An introduction to probabilistic graphical models, 2003.

[15] KDD Org. 2014 sigkdd test of time award. webpage, 2014. `http://www.kdd.org/News/view/2014-sigkdd-test-of-time-award`. Accessed 11.8.2016.

[16] Lloyd, S. Least squares quantization in pcm. *IEEE transactions on information theory 28*, 2 (1982), 129–137.

[17] Maaten, L. v. d., and Hinton, G. Visualizing data using t-sne. *Journal of Machine Learning Research 9*, Nov (2008), 2579–2605.

[18] MacQueen, J., et al. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability* (1967), vol. 1, Oakland, CA, USA., pp. 281–297.

[19] McLachlan, G., and Krishnan, T. *The EM algorithm and extensions*, vol. 382. John Wiley & Sons, 2007.

[20] Pearl, J. *Probabilistic reasoning in intelligent systems: networks of plausible inference.* Morgan Kaufmann, 2014.

[21] Rabiner, L. R. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE 77*, 2 (1989), 257–286.

[22] Wikipedia. DBSCAN — Wikipedia, the free encyclopedia. webpage, 2016. `https://en.wikipedia.org/wiki/DBSCAN`, Accessed 9.8.2916.

[23] Yianilos, P. N. Data structures and algorithms for nearest neighbor search in general metric spaces. In *SODA* (1993), vol. 93, pp. 311–21.

Appendix A

First appendix

Cluster No.	Samples
Cluster 0	945438 ENROLLING 0 WAITING 77 WAITING -1 1230357 ENROLLING 45 WAITING 13 WAITING -1 1960464 ENROLLING 3 WAITING 24 WAITING -1 14813553 ENROLLING 8 WAITING 33 WAITING -1 15253762 ENROLLING 1 WAITING 70 WAITING -1 15254301 ENROLLING 110 WAITING -1 16146920 ENROLLING 128 WAITING 4 IN_TREATMENT_ROOM 52 CLOSED -1 16335930 ENROLLING 1 WAITING 16 WAITING 5 WAITING 1 CANCELLED -1 16759935 ENROLLING 33 WAITING 55 WAITING -1 16760057 ENROLLING 0 WAITING 0 IN_TREATMENT_ROOM 53 CANCELLED -1
Cluster 1	8490996 ENROLLING 0 WAITING 71 WAITING 0 WAITING -1 13585666 ENROLLING 0 WAITING 78 WAITING -1 15636771 ENROLLING 2 WAITING 23 WAITING -1 16760229 ENROLLING 0 WAITING 76 WAITING -1 17383819 ENROLLING 106 WAITING 0 WAITING -1 17383955 ENROLLING 0 WAITING 18 WAITING 0 WAITING -1 18441848 ENROLLING 222 WAITING 0 WAITING -1 20080659 ENROLLING 0 WAITING 30 WAITING 0 WAITING -1 20330934 ENROLLING 97 WAITING 0 WAITING -1 20454780 WAITING 1 WAITING 239 WAITING 0 WAITING 0 WAITING -1
Cluster 2	15253761 ENROLLING 1 WAITING 27 IN_TREATMENT_ROOM 43 CLOSED -1 15833795 ENROLLING 28 WAITING 48 IN_TREATMENT_ROOM 6 CLOSED -1 15845553 ENROLLING 0 WAITING 16 IN_TREATMENT_ROOM 9 CLOSED -1 15856901 ENROLLING 1 WAITING 4 IN_TREATMENT_ROOM 4 CLOSED -1 15982252 ENROLLING 3 WAITING 2 IN_TREATMENT_ROOM 12 CLOSED -1 16080179 ENROLLING 1 WAITING 3 IN_TREATMENT_ROOM 5 CLOSED -1 16239380 ENROLLING 1 WAITING 1 IN_TREATMENT_ROOM 10 CLOSED -1 16240164 ENROLLING 0 WAITING 41 IN_TREATMENT_ROOM 13 CLOSED -1 16261761 ENROLLING 0 WAITING 1 IN_TREATMENT_ROOM 21 CLOSED -1 16285157 ENROLLING 22 WAITING 16 IN_TREATMENT_ROOM 4 CLOSED -1
Cluster 3	16270356 ENROLLING 1 WAITING 124 WAITING -1 16718646 ENROLLING 1533 CANCELLED -1 16802740 ENROLLING 480 CANCELLED -1 16803188 ENROLLING 35 CANCELLED -1 16808873 ENROLLING 1533 CANCELLED -1 16820174 ENROLLING 173 CANCELLED -1 16847486 ENROLLING 3 CANCELLED -1 16848514 ENROLLING 91 CANCELLED -1 16896527 ENROLLING 84 CANCELLED -1 16911990 ENROLLING 100 CANCELLED -1

Cluster 4	16759931 ENROLLING 1 WAITING 0 IN_TREATMENT_ROOM 89 CLOSED -1
	16759933 ENROLLING 12 WAITING 0 IN_TREATMENT_ROOM 35 CLOSED -1
	16759945 ENROLLING 93 WAITING 0 IN_TREATMENT_ROOM 5 CLOSED -1
	16760072 ENROLLING 33 WAITING 0 IN_TREATMENT_ROOM 5 CLOSED -1
	16760633 ENROLLING 8 WAITING 0 IN_TREATMENT_ROOM 45 CLOSED -1
	16760641 ENROLLING 24 WAITING 0 IN_TREATMENT_ROOM 2 CLOSED -1
	16802066 ENROLLING 1 WAITING 0 IN_TREATMENT_ROOM 19 CLOSED -1
	16802074 ENROLLING 3 WAITING 0 IN_TREATMENT_ROOM 5 CLOSED -1
	16802391 ENROLLING 1 WAITING 0 IN_TREATMENT_ROOM 22 CLOSED -1
	16802516 ENROLLING 2 WAITING 0 IN_TREATMENT_ROOM 23 CLOSED -1
Cluster 5	16760211 ENROLLING 0 WAITING 36 CANCELLED -1
	16802397 ENROLLING 0 WAITING 58 CANCELLED -1
	16802538 WAITING 49 CANCELLED -1
	16802716 ENROLLING 0 WAITING 2 CANCELLED -1
	16802724 ENROLLING 0 WAITING 9 CANCELLED -1
	16803196 ENROLLING 0 WAITING 2 CANCELLED -1
	16808893 ENROLLING 0 WAITING 0 WAITING 23 CANCELLED -1
	16824757 WAITING 3 CANCELLED -1
	16825742 WAITING 1 CANCELLED -1
	16841048 ENROLLING 0 WAITING 0 WAITING 2 CANCELLED -1
Cluster 6	16760216 ENROLLING 27 WAITING 92 IN_TREATMENT_ROOM 0 CLOSED -1
	16802089 ENROLLING 0 WAITING 92 IN_TREATMENT_ROOM 0 CLOSED -1
	16803170 ENROLLING 0 WAITING 17 IN_TREATMENT_ROOM 0 CLOSED -1
	16883234 WAITING 1 IN_TREATMENT_ROOM 0 CLOSED -1
	16889884 ENROLLING 0 WAITING 6 IN_TREATMENT_ROOM 0 CLOSED -1
	17137323 ENROLLING 2 WAITING 4 IN_TREATMENT_ROOM 0 CLOSED -1
	17144229 ENROLLING 2 WAITING 66 IN_TREATMENT_ROOM 0 CLOSED -1
	17266404 ENROLLING 34 WAITING 48 IN_TREATMENT_ROOM 0 CLOSED -1
	17266727 ENROLLING 44 WAITING 21 IN_TREATMENT_ROOM 0 CLOSED -1
	17383049 ENROLLING 0 WAITING 33 IN_TREATMENT_ROOM 0 CLOSED -1
Cluster 7	16760390 ENROLLING 0 WAITING 0 IN_TREATMENT_ROOM 13 CLOSED -1
	16801915 ENROLLING 0 WAITING 0 IN_TREATMENT_ROOM 7 CLOSED -1
	16802224 ENROLLING 0 WAITING 0 IN_TREATMENT_ROOM 8 CLOSED -1
	16802368 ENROLLING 195 WAITING 37 IN_TREATMENT_ROOM 2 WAITING 2929 ENROLLING 0 WAITING 0 IN_TREATMENT_ROOM 21 CLOSED -1
	16802390 ENROLLING 0 WAITING 0 IN_TREATMENT_ROOM 4 CLOSED -1
	16802750 ENROLLING 0 WAITING 0 IN_TREATMENT_ROOM 7 CLOSED -1
	16803047 ENROLLING 0 WAITING 0 IN_TREATMENT_ROOM 3 CLOSED -1
	16803314 ENROLLING 0 WAITING 0 IN_TREATMENT_ROOM 8 CLOSED -1
	16824756 ENROLLING 0 WAITING 0 IN_TREATMENT_ROOM 41 CLOSED -1
	16824867 ENROLLING 0 WAITING 0 IN_TREATMENT_ROOM 10 CLOSED -1
Cluster 8	16825520 ENROLLING 72 WAITING 7 IN_TREATMENT_ROOM 39 CLOSED -1
	16825742 ENROLLING 2 WAITING 0 CANCELLED -1
	16847801 ENROLLING 53 WAITING 1 CANCELLED -1
	16896665 ENROLLING 3 WAITING 8 WAITING 0 CANCELLED -1
	16898355 ENROLLING 50 WAITING 1 CANCELLED -1
	17149264 ENROLLING 11 WAITING 0 CANCELLED -1
	17266883 ENROLLING 1 WAITING 55 WAITING 0 CANCELLED -1
	17383018 ENROLLING 3 WAITING 113 IN_TREATMENT_ROOM 7 CLOSED -1
	17383043 ENROLLING 9 WAITING 42 IN_TREATMENT_ROOM 73 CLOSED -1
	17383842 ENROLLING 11 WAITING 1 CANCELLED -1
Cluster 9	16896538 ENROLLING 1 WAITING 32 CANCELLED -1
	16898635 ENROLLING 1 WAITING 6 CANCELLED -1
	17073845 ENROLLING 1 WAITING 5 IN_TREATMENT_ROOM 0 WAITING 6 IN_TREATMENT_ROOM 6 CLOSED -1
	17137406 ENROLLING 44 WAITING 3 IN_TREATMENT_ROOM 56 CLOSED -1
	17137437 ENROLLING 1 WAITING 56 CANCELLED -1
	17137811 ENROLLING 1 WAITING 18 CANCELLED -1
	17147824 ENROLLING 0 WAITING 1323 CANCELLED -1
	17250342 ENROLLING 1 WAITING 68 CANCELLED -1
	17383392 ENROLLING 42 WAITING 3 IN_TREATMENT_ROOM 50 CLOSED -1

	17383812 ENROLLING 1 WAITING 7 CANCELLED -1
Cluster 10	16896542 WAITING 0 WAITING 4 WAITING -1 16898754 ENROLLING 10 WAITING 55 IN_TREATMENT_ROOM 2 CLOSED -1 16911970 WAITING 239 CANCELLED -1 18042974 ENROLLING 8 WAITING 1 IN_TREATMENT_ROOM 71 CLOSED -1 18206521 ENROLLING 0 WAITING 0 WAITING 136 CANCELLED -1 18206522 ENROLLING 0 WAITING 0 WAITING 136 CANCELLED -1 19336998 ENROLLING 0 WAITING 1 IN_TREATMENT_ROOM 21 WAITING 0 WAITING 536 CANCELLED -1 19488319 WAITING 13 WAITING 0 WAITING 0 CANCELLED -1 19517799 WAITING 215 CANCELLED -1 19551936 ENROLLING 3 WAITING 0 WAITING 154 CANCELLED -1
Cluster 11	17141316 ENROLLING 3 WAITING 24 CANCELLED -1 17267028 ENROLLING 59 WAITING 13 IN_TREATMENT_ROOM 69 CLOSED -1 17333621 ENROLLING 1 WAITING 1 CANCELLED -1 17384117 ENROLLING 40 WAITING 11 IN_TREATMENT_ROOM 52 CLOSED -1 17384134 ENROLLING 1 WAITING 105 IN_TREATMENT_ROOM 92 CLOSED -1 17445733 ENROLLING 1104 ENROLLING -1 17472038 ENROLLING 9 WAITING 37 IN_TREATMENT_ROOM 70 CLOSED -1 17472501 ENROLLING 52 WAITING 118 IN_TREATMENT_ROOM 22 CLOSED -1 17484544 ENROLLING 0 WAITING 1457 ENROLLING 0 WAITING -1 17512538 ENROLLING 0 WAITING 16 WAITING 1002 CANCELLED -1

Table A.1: 10 samples from each cluster formed by DBSCAN

log-likelihood	sample
-1.22501248649	21332189 ENROLLING 0 WAITING 3 IN_TREATMENT_ROOM 5 CLOSED -1
-1.22501248649	21332152 ENROLLING 0 WAITING 3 IN_TREATMENT_ROOM 5 CLOSED -1
-1.22501248649	21243121 ENROLLING 0 WAITING 3 IN_TREATMENT_ROOM 5 CLOSED -1
-1.22501248649	21242603 ENROLLING 0 WAITING 3 IN_TREATMENT_ROOM 5 CLOSED -1
-1.22501248649	20753590 ENROLLING 0 WAITING 3 IN_TREATMENT_ROOM 5 CLOSED -1
-1.31252682278	20531975 ENROLLING 0 WAITING 9 IN_TREATMENT_ROOM 5 CLOSED -1
-1.31252682278	20335903 ENROLLING 0 WAITING 9 IN_TREATMENT_ROOM 5 CLOSED -1
-1.31252682278	20332176 ENROLLING 0 WAITING 9 IN_TREATMENT_ROOM 5 CLOSED -1
-1.31252682278	20330946 ENROLLING 0 WAITING 9 IN_TREATMENT_ROOM 5 CLOSED -1
-1.31252682278	19524600 ENROLLING 0 WAITING 9 IN_TREATMENT_ROOM 5 CLOSED -1
-1.31252682278	18895611 ENROLLING 0 WAITING 9 IN_TREATMENT_ROOM 5 CLOSED -1
-1.6200398225	20090272 ENROLLING 0 WAITING 20 IN_TREATMENT_ROOM 5 CLOSED -1
-1.6200398225	17137238 ENROLLING 0 WAITING 20 IN_TREATMENT_ROOM 5 CLOSED -1
-1.6200398225	17137111 ENROLLING 0 WAITING 20 IN_TREATMENT_ROOM 5 CLOSED -1
-1.6200398225	16803024 ENROLLING 0 WAITING 20 IN_TREATMENT_ROOM 5 CLOSED -1
-1.62051607252	21419996 ENROLLING 0 WAITING 14 IN_TREATMENT_ROOM 13 CLOSED -1
-1.81955956629	18946307 ENROLLING 1 WAITING 0 IN_TREATMENT_ROOM 10 CLOSED -1
-1.81955956629	17010037 ENROLLING 1 WAITING 0 IN_TREATMENT_ROOM 10 CLOSED -1
-1.81974417979	20538959 ENROLLING 1 WAITING 5 IN_TREATMENT_ROOM 13 CLOSED -1
-1.81974417979	16824641 ENROLLING 1 WAITING 5 IN_TREATMENT_ROOM 13 CLOSED -1
-1.82019772133	19969197 ENROLLING 2 WAITING 7 IN_TREATMENT_ROOM 6 CLOSED -1
-2.09969260165	21332560 ENROLLING 6 WAITING 4 IN_TREATMENT_ROOM 8 CLOSED -1
-2.09974363796	21419663 ENROLLING 1 WAITING 1 IN_TREATMENT_ROOM 19 CLOSED -1
-2.10057254235	19664612 ENROLLING 1 WAITING 19 IN_TREATMENT_ROOM 11 CLOSED -1
-2.10063359561	17267455 ENROLLING 3 WAITING 9 IN_TREATMENT_ROOM 12 CLOSED -1
-2.10077419383	20538310 ENROLLING 1 WAITING 20 IN_TREATMENT_ROOM 10 CLOSED -1
-2.52467325597	18335666 ENROLLING 1 WAITING 33 IN_TREATMENT_ROOM 10 CLOSED -1
-2.52507176943	18243950 ENROLLING 1 WAITING 35 IN_TREATMENT_ROOM 4 CLOSED -1
-2.52507176943	16911955 ENROLLING 1 WAITING 35 IN_TREATMENT_ROOM 4 CLOSED -1
-2.52523586276	18160697 ENROLLING 2 WAITING 31 IN_TREATMENT_ROOM 5 CLOSED -1
-2.52574525145	19841457 ENROLLING 7 WAITING 20 IN_TREATMENT_ROOM 7 CLOSED -1
-3.06107298199	19954450 ENROLLING 0 WAITING 61 IN_TREATMENT_ROOM 3 CLOSED -1
-3.06260151935	21127113 ENROLLING 11 WAITING 30 IN_TREATMENT_ROOM 9 CLOSED -1
-3.06387434043	19230307 ENROLLING 28 WAITING 10 IN_TREATMENT_ROOM 11 CLOSED -1
-3.06458227144	18247745 ENROLLING 2 WAITING 10 IN_TREATMENT_ROOM 33 CLOSED -1
-3.06629701355	17141536 ENROLLING 26 WAITING 13 IN_TREATMENT_ROOM 2 CLOSED -1
-3.77652544852	20624690 ENROLLING 7 WAITING 54 IN_TREATMENT_ROOM 11 CLOSED -1
-3.77716200844	19230163 ENROLLING 0 WAITING 32 IN_TREATMENT_ROOM 43 CLOSED -1
-3.77716858350	17266875 ENROLLING 3 WAITING 25 IN_TREATMENT_ROOM 36 CLOSED -1
-3.77756155251	16848287 ENROLLING 58 WAITING 10 IN_TREATMENT_ROOM 8 CLOSED -1
-3.77963738529	17384149 ENROLLING 1 WAITING 42 CANCELLED -1
-5.10571464997	20032830 ENROLLING 28 CANCELLED -1
-5.10571464997	18004184 ENROLLING 28 CANCELLED -1
-5.10571464997	18004183 ENROLLING 28 CANCELLED -1
-5.10766253514	18945732 ENROLLING 30 WAITING 11 IN_TREATMENT_ROOM 48 CLOSED -1
-5.10842170221	18043295 ENROLLING 0 WAITING 19 IN_TREATMENT_ROOM 71 CLOSED -1
-5.11050052549	20538023 ENROLLING 60 WAITING 8 CANCELLED -1
-227.515556821	18883776 ENROLLING 0 WAITING 4435 ENROLLING -1
-294.647987389	19593547 ENROLLING 0 WAITING 5754 ENROLLING -1
-653.392679598	19229957 ENROLLING 37265 ENROLLING 40 WAITING 1 IN_TREATMENT_ROOM 4 CLOSED -1
-876.235768456	21127118 ENROLLING 50174 ENROLLING 0 WAITING 1 IN_TREATMENT_ROOM 3 CLOSED -1
-1227.10081314	17144893 ENROLLING 0 WAITING 23 WAITING 48064 ENROLLING 0 WAITING 18 CANCELLED -1

Table A.2: Samples with log-likelihood computed using first-order Markov Chain using negative binomial distribution as emission function.