Modeling User Behavior from E-Commerce Data with Hidden Markov Models and Logistic Regression

Nakisa Mohammadifard

Master of Science

School of Computer Science

McGill University

Montreal,Quebec

2013-03-21

A thesis submitted to McGill University in partial fulfillment of the requirements for the degree of Master of Science

DEDICATION

To my parents

ACKNOWLEDGEMENTS

I would like to thank everybody who guided and supported me through this thesis. I specially thank my supervisor, DoinaPrecup, for her valuable guides and endless understanding and support. I would like to thank all members of McGills Reasoning and Learning lab specially Negar and Amir-Masoud for the useful discussions on the thesis.I also like to thank my friends Shahin, Danesh and Atousa who helped and supported me during this time. Finally, I would like to thank my family for encouraging me to pursue graduate studies and supporting me unconditionally through this way.

ABSTRACT

Mining online user data has become more and more popular in e-commerce. Businesses are using this data to do customization and user behavior prediction and one main applications of this information is advertisement. In this thesis, we address the problem of finding the influence of advertisements on a user's purchase behavior, by using machine learning methods to analyze purchase data obtained from real online retail systems. The hypothesis driving the model we propose is that different ads have different influences, but also the same ad can make the user behave differently if she is in different inner states. To capture this last aspect, we approached this problem using Hidden Markov Models for users. To consider the influence of ads and their properties, we replaced the traditional observation model of a Hidden Markov Model with Logistic Regression, which allows us to define an observation model depending not only on the HMM state, but also on external events such as advertising campaigns. We use a large online user data by an industry partner and our model is fit to predict if the user will make a purchase at a specific time interval or not.

ABRÉGÉ

L'éxploration des données utilisateurs en ligne est devenu de plus en plus populaire dans le commerce éléctronique. Les entreprises utilisent ces données pour la personnalisation et la prévision du comportement des utilisateurs, et une des applications principales de cette information est la publicité. Dans cette thèse, nous abordons le problème de trouver l'influence de la publicité sur le comportement d'achat d'un utilisateur, à l'aide de l'apprentissage automatique. Nous utilisons un jeu de données réelles provenant d'un détaillant en ligne. L'hypothèse de base du modèle que nous proposons est que les différentes annonces ont des influences différentes, mais aussi la meme annonce peut faire l'utilisateur se comporter différemment si lui ou elle se trouve dans différents états intérieurs. Pour capturer ce dernier aspect, nous avons abordé ce problème en utilisant des modèles Markov cachés pour modeller les utilisateurs. Pour étudier l'influence des annonces et leurs propriétés, nous avons remplacé le modèle d'observation traditionnelle par modèle Markov cach$^{\acute{}}$ avec la régression logistique, ce qui nous permet de définir un modèle d'observation en fonction non seulement de l'état HMM, mais aussi sur des événements externes tels que les campagnes de publicité. Nous utilisons une grande quantité de données utilisateur en ligne provenant d'un partenaire industriel. Notre modèle est ajusté pour prédire si l'utilisateur fera un achat dans un intervalle de temps spécifié ou non.

TABLE OF CONTENTS

LIST OF TABLES

Table page

LIST OF FIGURES

CHAPTER 1
Introduction

To predict the behavior of users on web has become very popular and also important for online businesses. More and more businesses are now collecting the log of users interactions with their website, as well as personal information whenever possible, and do analysis on this data in order to find ways to boost their competitive advantage. One way of using this data is to provide better interactions with users via customization of both content and delivery of information, whether through web pages, emails or other forms of communication. Mining this data also has the advantage of providing a basis for more reliable business decisions and revenue estimation.

One of the most popular applications of online data mining in online business is advertising. Companies are trying to customize advertisement for users based on their characteristics, online behavior, history of ads shown to them and their reaction, etc. The goal is to determine which advertisements have the highest probability to generate a purchase from a specific user and show those to her. When users are anonymous or the use of personal information is a privacy matter, it is still possible to use the online behavior of a large set of users (for example click data) to show better advertisement to them. Many large companies, such as Google or Yahoo, use data mining to optimize advertising, but this also happens with smaller online retail companies.

Customizing ads for users is one side of the story and the other side of it is to find how valuable an advertisement is. Holding an advertising event has costs and if companies get an estimate of the marginal impression value of an ad, they can decide whether to run an event or not, and what type of event would be most beneficial. This estimate of an ad impression value is attainable through the aggregation of the reactions of many users to similar ads.

Our goal in this research was to predict user behavior towards advertisements, which helps both with estimating the marginal impression value of an advertisement and with ad customization. The behavior of a user who receives an advertisement depends both on the properties of the advertisement, as well as on the satisfaction level or inner state of the user towards that e-business based on her past purchase experience as well as based on word of mouth. The same advertisement can affect a new user, a satisfied old user, and an unsatisfied angry user very differently. Information about the state of the user can also be of benefit by using in the prediction users with similar states or characteristics. However, the challenge is that we do not necessarily know the internal state of the user, especially when users are anonymized for privacy reasons or the business does not maintain detailed user profiles. Hence, we use machine learning methods to leverage large amounts of data provided by a partner company[1] and solve this problem.

[1] The company cannot be named due to a non-disclosure agreement

1.1 Outline of methodology

For this project, we were lucky to have access to two years worth of purchase history data and advertisement events data from an online business (which cannot be named due to a non-disclosure agreement). Working with user online behavior data is challenging because there are many hidden factors which influence the user purchase behavior. Also, there is usually a lot of variation between the start points of the histories of different users and therefore between the lengths of different times series. Overall, the numbers of purchases is very low in comparison to time frame of data and usually there are plenty of one-time users. The other difficulty in working with the data we had was the discrepancy of the time frames of purchase history data and the advertisement events. More particularly, if we see a user purchasing a product, we do not know for sure if this purchase was due to a recent advertisement they saw, a past advertisement, or just an immediate need (like a birthday) that has nothing to do with an advertisement. At the same time, working with real data is important in order to test in depth how well machine learning algorithms perform in challenging situations.

To capture the influence of both advertisement properties and user inner state on user purchase behaviour prediction, we propose to use a customized probabilistic model for this problem. The model is built using a transition structure like a Hidden Markov Model (HMM) model. However, instead of using usual probability distributions for the observation model, we use a Logistic Regression, whose output should predict the probability of a purchase given the current user state and advertisement information. We chose the HMM internal structure to be able to model different user

inner states which are hidden and transitions of the user between these inner states, and the Logistic Regression observation model is for incorporating the information we have about advertisement properties. We use one Logistic Regression model for each inner state and therefore for users in the same states, an advertisement will cause in same probability of purchase results.

1.2 Thesis outline

This thesis starts with a background chapter on Hidden Markov Models and Logistic Regression and also related work on applications of machine learning, especially HMMs, in e-Commerce and user behavior prediction. In Chapter 3, we describe the data used and preprocessing needed, our hybrid model, and the algorithm we use to train the logistic regression HMM we propose. Chapter 4 details the experimental setup and results of the algorithms we used. We ran experiments both on simulated data, to ensure a proper checking and validation of the proposed hybrid model, as well as on real data. Finally, the last chapter contains conclusions and discusses future work.

CHAPTER 2
Background

This chapter consists of theoretical background on Hidden Markov Models, Logistic Regression and related methods, and a review work in the applications of Hidden Markov Models in e-commerce and e-business.

2.1 Hidden Markov Models

Hidden Markov Models (HMMs) are an extension of Markov Chains in which the state of the chain is not directly available and there are only imprecise stochastic observations that provide information about the state. Hidden Markov Models can be used to model broader sets of problems than Markov Models since they do not assume we know the states, which is more similar to what we encounter in the real world. We now review the main concepts and algorithms related to HMMs, based on the classic tutorial by Rabiner [8].

An HMM consists of N states. At every discrete time t, the process is in some state. Due to the Markovian property, the state at time step t is determined stochastically based on the state at the previous time step, $t-1$, but does not depend on any states before $t-1$. There is a transition probability distribution which relates the previous state to the current one. There is an observation O_t at each time step which depends on the state Q_t. For every possible state, there is a probability distribution which generates the observations.

We use the following notation:

- T: number of time steps in a sequence of observations
- N: number of possible states
- M: number of possible observations
- S: set of N possible states
- V: set of M possible observations
- Q_t: random variable denoting the state at time step t
- O_t: random variable denoting the observation at time step T
- A: transition probability matrix of size $N \times N$, where $a_{ij} = P(Q_{t+1} = j | Q_t = i)$ is the probability of transitioning from the ith state to the jth state, with i and j between 1! and N
- B: observation probability matrix of size $N \times M$, where $b_{jk} = P(O_t = k | Q_t = j)$ is the probability of the kth observation in the jth state, with $k = 1 \dots M$ and $j = 1 \dots N$
- π: initial state distribution, which is a vector of size N, with $\pi_i = P(Q_1 = i)$.
- $O = O_1 \dots O_T$: observations sequence
- $Q = Q_1 \dots Q_t$: state sequence

The HMM consists of the tuple $\lambda = \langle A, B\pi \rangle$. An HMM generates observations as follows. For time step $t = 1$, the initial state Q_1 is drawn from the initial state distribution π. In each time step t, the observation O_t is generated by the observation probability distribution for state Q_t, according to the corresponding row of B. . States for each time step $t > 1$ are sampled from the transition probability distribution for the previous state, as given in the corresponding row of matrix A

There are three basic problems for HMMs. The first problem is finding the observation sequence probability $P(O)$. The second problem is finding the most probable state sequence for the seen observation sequence: $\arg\max_Q P(Q|O)$. The third problem is choosing model parameters that maximize the probability of a set of observation sequences.

2.1.1 Finding the probability of a state sequence

Finding the probability of an observation sequence O in an HMM with model λ can be done naively by considering all possible sequences of states Q, computing $P(O, Q)$ and marginalizing over Q:

$$P(O|\lambda) = \sum_Q P(O, Q|\lambda) = \sum_{Q_1,\ldots Q_T} \pi_{Q_1} b_{Q_1 O_1} a_{Q_1 Q_2} b_{Q_2 O_2} \ldots a_{Q_{T-1} Q_T} b_{Q_T O_T}$$

There are N^T possible state sequences since we have T time steps and N possible states at each time. In addition, for every fixed state sequence we have T computations to perform so the overall calculation cost will be TN^T. This is not feasible even for relatively small values of T and N. Instead of the above approach, a fairly efficient alternative dynamic programming algorithm called the forward-backward algorithm is used. In this algorithm, we define the probability of the partial observation from time step 1 to t and being in state i at time step as $\alpha_t(i)$:

$$\alpha_t(i) = P(O_1, \ldots O_t, Q_t = i|\lambda)$$

These probabilities can be computed recursively as follows:

$$\alpha_1(i) \quad = \quad \pi_i b_{iO_1}, 1 \leq i \leq N$$

$$\alpha_{t+1}(j) = \left[\sum_{i=1}^{N} \alpha_t(i) a_{ij}\right] b_{jO_{t+1}}, 1 \leq t \leq T-1$$

Finally, to get the observation sequence probability, we just need to sum $\alpha_T(i)$ over all possible states:

$$P(O|\lambda) = \sum_{i=1}^{N} \alpha_T(i) \tag{2.1}$$

Hence, the result is computed in time $O(TN^2)$, which is considerably faster than using the naive algorithm.

A similar computation can also be done backward. Let $\beta_t(i)$ be the probability of the partial observation sequence between $t+1$ and T conditioned on being in state i at time step t:

$$\beta_t(i) = P(O_{t+1}O_{t+2}\ldots O_t|Q_t = i, \lambda)$$

Similarly to the αs, the βs can be computed recursively, going backwards in time:

$$\begin{aligned} \beta_T(i) &= 1, 1 \leq i \leq N \\ \beta_t(i) &= \sum_{j=1}^{N} a_{ij} b_{jO_{t+1}} \beta_{t+1}(j), 1 \leq i \leq N, 1 \leq t \leq T-1 \end{aligned}$$

The backward computation is not needed for computing the probability of an observation sequence, but it is useful for solving the other two HMM problems.

2.1.2 Finding the most probable state sequence

In this section we discuss how to find the optimal sequence of states associated with an observation sequence. The most likely state sequence is the one which maximizes $P(Q|O, \lambda)$, which is equivalent (using Bayes rule) to maximizing $P(Q, O|\lambda)$. The optimal state sequence under this definition can be obtained using the Viterbi algorithm, which is a dynamic programming algorithm.

Let $\delta_t(i)$ be the highest probability of being in state i at time step t:

$$\delta_t(i) = \max_{Q_1,\ldots Q_{t-1}} P(Q_1,\ldots Q_{t-1}, Q_t = i, O_1,\ldots O_{t-1}, O_t|\lambda)$$

We can write the δs recursively as:

$$\delta_{t+1}(j) = \left[\max_i \delta_t(i) a_{ij}\right] b_{jO_{t+1}}$$

We need to keep track of the states j in each time step. To store these values, we use an array named $\Psi_t(j)$.

The Viterbi algorithm starts with the following initialization, for all $1 \leq i \leq N$:

$$\begin{aligned} \delta_1(i) &= \pi_i b_{iO_1} \\ \Psi_1(i) &= 0 \end{aligned}$$

The the algorithm proceeds by applying the following recursive equations for all $1 \leq t]leqT$ and $1 \leq j \leq N$:

$$\begin{aligned} \delta_t(j) &= \left[\max_i \delta_{t-1}(i) a_{ij}\right] b_{jO_t} \\ \Psi_t(j) &= \arg\max_i \left[\delta_{t-1}(i) a_{ij}\right] \end{aligned}$$

Once these are computed, to re-build the path we do backtracking starting at T. Let Q_t^* be the most probable state at time step t. Then:

$$Q_T^* = \arg\max_i \delta_t(i) \tag{2.2}$$

$$Q_t^* = \Psi_{t+1}(Q_{t+1}^*) \tag{2.3}$$

The probability of the maximum state sequence is given by: $P^* = \max_i \delta_T(i)$.

2.1.3 Finding the HMM model parameters

Finding the HMM model parameters is the main problem of the HMM. Here we want to identify a model that yields a high probability for the seen observation sequences. Known methods are mostly iterative and find locally optimal parameters. Our focus is on the iterative Baum-Welch algorithm, which is an expectation-maximization-type method.

In this algorithm we start with initial HMM model parameters and iteratively update and improve them. Let $\xi_t(i,j)$ be the probability of being in state i at time step t and in state j at time step $t+1$ given an observation sequence O and model parameters λ:

$$\xi_t(i,j) = P(Q_t = i, Q_{t+1} = j|O, \lambda)$$

We can write $\xi_t(i,j)$ as a function of the α and β parameters defined in Section 2.1.1:

$$\xi_t(i,j) = \frac{\alpha_t(i)a_{ij}b_{jO_{t+1}}\beta_{t+1}(j)}{P(O|\lambda)} = \frac{\alpha_t(i)a_{ij}b_{jO_{t+1}}\beta_{t+1}(j)}{\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_t(i)a_{ij}b_{jO_{t+1}}\beta_{t+1}(j)}$$

Note that $\sum_{t=1}^{T-1}\xi_t(i,j)$ is the expected number of transitions from state i to state j over the entire sequence.

Let $\gamma_t(i)$ be the probability of being in state i at time t, given the observation sequence O and the model λ:

$$\gamma_t(i) = P(Q_t = i|O, \lambda)$$

Note that $\gamma_t(i)$ can also be expressed as a function of the αs and βs as:

$$\gamma_t(i) = \frac{\alpha_t(i)\beta_t(i)}{P(O|\lambda)} = \frac{\alpha_t(i)\beta_t(i)}{\sum_{i=1}^{N}\alpha_t(i)\beta_t(i)}$$

Note that $\sum_{t=1}^{T-1} \gamma_t(i)$ is the expected total number of transitions out of state i over the sequence of data under consideration. Now we can use these quantities to explain how to re-estimate the model parameters in order to get a better fit to the data.

The Baum-Welch algorithm starts with an initial guess for the model parameters λ. In each successive iteration, these parameters are re-computed as follows. The initial state probabilities are given by the expected number of times for being in each state at the initial time step:

$$\hat{\pi}_i = \gamma_1(i), \forall 1 \leq i \leq N$$

(we use $\hat{.}$ to denote parameter values that are computed from data, and hence which approximate the true model).

The transition model is computed (using the previously described symbols) as:

$$\hat{a}_{ij} = \frac{\sum_{t=1}^{T-1} \xi_t(i,j)}{\sum_{t=1}^{T-1} \gamma_t(i)}$$

The observation model $\hat{b}_{ik}$ is given by the expected number of times when the system is in state i and the observation is k, divided by the number of times in state j:

$$\hat{b}_{ik} = \frac{\sum_{t \in \{1,\ldots T\}: O_t = k} \gamma_t(i)}{\sum_{t=1}^{T} \gamma_t(i)}$$

It can be shown that the new model either improves the likelihood of the provided data, or leaves it the same as for the previous model. This iterative procedure converges to a critical point of the log-likelihood function, and looking at the second derivative of this function ensures this is a local maximum. However, in general

this is only a locally optimal solution. In practice, random re-starts from different parameter values are used to make sure that a good solution is found in the end.

The type of distribution used for the HMM parameters depends on the type of data to be handled. Most HMMs assume discrete states, in which case the A matrix contains multinomial distributions. If the observations are also discrete, multinomials can be used for the B matrix as well. However, if the observations are continuous, the observation probabilities would not be modelled through a matrix, but through different distributions for each state, such as Gaussians or mixture of Gaussians. However, the main steps of the EM algorithm remain the same in this case.

2.2 Logistic regression

We now briefly explain logistic regression, which will be used in our experiments to model observation emissions. Logistic regression relies on the logistic or sigmoid function, defined as:

$$\sigma(a) = \frac{1}{1 + e^{-a}}$$

This yields an S-shaped curve. For any value of a, the output of the function is between 0 and 1, which gives it a natural probabilistic interpretation. The logistic function also has a symmetry property:

$$\sigma(-1) = 1 - \sigma(a)$$

The inverse of the logistic function, named the logit function, is given by:

$$a = \ln\left(\frac{\sigma}{1 - \sigma}\right)$$

Assuming that we want to solve a binary classification problem with classes 0 and 1, we can represent the conditional probability of an example being of class 1, as a logistic function of a linear combination over a feature vector ϕ:

$$P(y = 1|\phi) = \sigma(\mathbf{w}^T\phi) = \frac{1}{1 + e^{-\mathbf{w}^T\phi}}$$

The goal of the learning is to find a parameter vector $\mathbf{w}$ which, give a data set $(\phi_1, y_1) \ldots (\phi_n, y_n)$ maximizes the likelihood of the data:

$$\prod_{i=1}^{n} P(y = 1|\phi_i)^{y_i}(1 - P(y = 1|\phi_i)^{1-y_i}$$

Maximizing this likelihood can be achieved by minimizing the negative log-likelihood, also known as the cross-entropy function:

$$E(\mathbf{w}) = -\sum_{i=1}^{n} (y_i \log(P(y = 1|\phi_i) + (1 - y_i) \log(P(y = 1|\phi_1)))$$

This function has a unique minimum, but there is no closed-form way of computing it. One way to obtain the solution is to use gradient descent, updating the weight vector as:

$$\mathbf{w} \leftarrow \mathbf{w} + \epsilon \sum_{i=1}^{n} (y_i - P(y = 1|\phi_i)\phi_i$$

where $\epsilon \in (0, 1)$ is a step size or learning rate parameter.

A more direct way of solving the optimization, which does not require any parameters, is to use the Newton-Raphson method, which uses a local quadratic approximation to the log-likelihood function. It updates the parameter vector $\mathbf{w}$ as:

$$\mathbf{w} \leftarrow \mathbf{w} - H^{-1}\nabla E(\mathbf{w})$$

where H is the Hessian matrix which contains the second-order derivatives of the error function:

$$H = \sum_{i=1}^{n} y_i(1 - y_i)\phi_i\phi_i^T = -\Phi^T R\Phi$$

where Φ is a matrix containing all the feature vectors and R is a matrix with elements of the form $P(y = 1|\phi_i)(1 - (y = 1|\phi_i)$.

Newtons method usually takes fewer iterations than gradient descent and there is no learning rate parameter to adjust. On the other hand, computing Hessian matrix is time consuming and Newtons method may not be a good candidate for an on-line algorithm.

2.3 Related work

In this part we will go through the different applications of machine learning in e-commerce and we will specifically look for the applications of hidden Markov models. Finally we will introduce some of the related work on the application of machine learning techniques, especially hidden Markov models, on user behavior modelling and prediction.

2.3.1 Machine learning in e-commerce

As online data grows with an extremely fast pace, Machine Learning (ML) techniques are considered more and more in automatic data analysis for a variety of application domains. Many e-commerce tasks use machine learning techniques, among them recommender systems, fraud detection, data usage mining, and user behavior modeling.

Recommender systems are the widest usage of machine learning in ecommerce. Most of the e-businesses use recommenders to improve their consumer convenience.

Some of the famous e-businesses using the recommenders are Amazon.com, eBay, CDNow, MovieFinder.com, Reel.com. Automatic recommender systems are machine learning systems specialized to recommend products in commerce applications. These systems use a wide range of techniques, ranging from nearest neighbor to Bayesian analysis. Common data types used in recommender systems are demographic data, rating data, behavior pattern data, transaction data, and production data. The most successful recommendation technology is the collaborative filtering approach in which the key step is to find similar customers for the active customer and recommend based on the preferences of the similar customers, like the work by Resnick [9], applying K Nearest Neighbours to identify similar customers. On the other hand, there is also content based filtering approach like a work by Ono and Kurokawa [7],which applies Bayesian Network for a movie recommender system based on movie contents.

The other main application of machine learning is data usage mining which enables discovery of user patterns in the web usage. This information can be used for personalization, network traffic flow analysis, adaptive websites, business and support services, and website management. In [1], ant colony clustering is used to discover the usage patterns and linear genetic programming for the pattern analysis. In [4], Probabilistic Latent Semantic Analysis (PLSA) is used to uncover the semantic associations among users and pages based on co-occurrence patterns of these pages in user sessions.

2.3.2 HMMs in e-commerce

Hidden Markov models have been applied to a wide range of problems in e-Commerce. For example, in recommender systems, HMMs have been used in dynamic collaborative filtering for an article recommendation system [10]. In this work, a Negative Binomial mixture of Multinomial distributions is used as the observation model, in order to account for the possibility of change in the reader's preferences over time. Their work outperformed static algorithms and the matrix factorization based dynamic algorithm.

There is also work on evaluating the recommenders reputation using an HMM-based model [12] for the recommendation network in distributed trust systems. This approach models chained recommendation events as an HMM and the state transition matrix contains the probabilities that recommenders send the request to other recommenders.

Additionally, in web usage mining, HMM and Fuzzy clustering, which tries to measure similarity among the users based on their browsing characteristics, and predicts user behavior to achieve pre-fetching. They accomplished the result of 90% for similarity access and 88% pre-fetching accuracy using 3% cache [11].

2.3.3 Machine learning for user purchase behavior modelling and prediction

There are some applications and approaches of machine learning techniques including HMMs which directly target user purchase behavior modeling and prediction. Our application area of interest in this thesis concentrates on this target. To mention some of the machine learning approaches on user purchase behavior modeling and prediction, we discuss first a paper for applying relational Bayesian models for

modeling online user behavior [3]. They applied their method on user click stream sessions and got results of 71% with the AUC (area under the curve) measure for the prediction of last page in a session. Another paper on this topic [2] tries to classify online users in two categories of buy and not buy at each time step for the whole trajectory which is non-reversible, or the option of waiting for reveal of more information. Their methodology has a Bayesian scheme for estimating purchase probabilities. With defining a performance measure including the cost for the time of the prediction, they have demonstrated the classifiers using second order Markov models worked the best among the others.

There are two other works with very similar problem natures and taken approaches to ours that we discuss them in more detail. First is the work in [5], which employs the user log data of the web server of DIGICAKE (an on-line cake-ordering e-shop in Taiwan) and is preprocessed to a time series of individual user sessions, a path of specific user clicks through different pages. The hidden Markov model they use consist of a set of two states, where is the state of having the intention of purchase, is the state of not having the intention of purchase, and clicked pages are the observations. They use Baum Welch to get the model parameters starting with random values. After that they use Viterbi algorithm to get the most probable hidden states using the model parameters resulted from Baum Welch algorithm. They defined two measures for evaluating their model, precision and recall defined as below, and they got 51% precision and 73% recall accuracy for the purchase prediction of the users.

Also very related is the work in [13]. They used data on the customer movement path or shopping-path on different store areas in a supermarket, dividing the entire store into 16 different areas, collected by using carts with radio frequency identification (RFID) tags and tracking them at one-second intervals from entrance to the cash register. Shopping-path data has been considered as the time series data of sales areas with the stationary time (sec) the customer spent in those areas. There is also point of sale data with the areas in which items are sold. Customer behavior is modeled by the hidden Markov model. There are two possible states, , and the stationary times on sale areas was considered as the observed variable. They have assumed normal distributions for observation probabilities () and no initial and is provided in this paper. They have used Baum Welch algorithm to get the model parameters, and they used Viterbi algorithm to get the most probable hidden states using the resulting model parameters. They constructed an HMM model for each user independently and tried to find the most probable intermediate and terminating states. By assuming the stop state as purchase and pass by state as not purchase, they tried to predict user purchase behavior and they evaluated their method using recall, precision, and F-measure based on the POS data. They demonstrated average results for HMM and also for EM clustering [6], claiming that EM clustering is superior in recall and the HMM is superior in precision and they are similar in F-measure.

CHAPTER 3
Data and methods

In this chapter we describe the data we used for this project and the algorithms we employed to process it.

3.1 Data description

The project we worked on was defined through a collaboration between McGill University and a predictive analytics company. The company provided us with two years of data regarding customers that interacted with a large online retailer. The customers received advertisement for special sale events and made purchases. The goal of the project was to develop a model able to predict users future behavior.

The data provided to us includes four main categories of information: members, orders, products, and events. We now look in detail at each type of information. Members data includes users information: id, gender, location, date at which the user first interacted with the web site, and date of the last login. Orders data includes id, date when the order was placed, ids of the items that were ordered, order price, tax amount, shipping cost, billing city, and other order information (which is irrelevant for this work). The products information contains id, brand, category, price, as well as a text description of each product. We did not use the product information directly in this project. That is, we only consider dates and whether a user has made a purchase or not, but not the amount of the purchase or the product information.

This is a simplification, but it allows us to avoid processing the unstructured text information about the nature of each product.

Finally, the event data contains information about special promotions that were advertised to users, typically by email. Each event has a start and end date, as well as a text description of the sale information. Because it is quite difficult to deal with the text data, the company preprocessed the text descriptions and re-encoded events using 100 numerical features, obtained by a dimensionality reduction step performed on the text descriptions. We only use these features of the events (along with the dates) in our work.

In order to model the data, we need to generate a time series for each user. To do this, we combined the member and order categories to organize the data in such a way that we have time steps of one week in length and a corresponding time series of purchase activities for each user. From the members data, we found the minimum (earliest) time of the "date joined column and set the corresponding week as the earliest start of our time series ($t = 1$), the we found the maximum (latest) time of the "last login column and set the corresponding week as the end of our time series, which turned out to be $T = 63$. Hence, we will get a time series of 63 weeks for the oldest user on the site and time series shorter or equal to it for the other users.

We dont have any un-subscription information of users and therefore we will assume time series of all users end at T, but we have information on the dates at which users joined the site, and not all of them start at $t = 1$. The time series value for user u in time step t, $O_t(u)$, is a binary variable equal to 1 if u purchased anything in week t and 0 otherwise. The site had a total of over 7000 users, but

in most cases users made very few purchases during this time. In fact, a third of users made only one purchase, which is not adequate for fitting a time series model. In order to have time series that are long enough to process with success, from all users, we just picked those with more than 5 overall purchases to use in the work. The number of those users is $NU = 2029$.

The events data contains a total of 815 special advertisement events, which are the same for all users. The events last more than one time step, and they may be overlapping (so more than one event is going on at the same time). As explained above, each event is described by 1000 numerical features, with magnitude in the interval $[-1, 1]$. For time step t, we denote by I_t the set of events happening at t. This set contains the features vectors for all events active at that time.

3.2 Proposed methodology

As described above, for each user we generated a time series $O_t(u)$ which indicates if within each time period (week) the user purchased something or not. The goal of the learning is to compute a model which can predict this type of behavior for new users based on their history of purchasing and the events happening at the time.

Intuitively, purchase activities are dependent on the users inner states. For instance, a satisfied user is more likely to buy a product than an unsatisfied or a new unfamiliar user. To capture this idea, we defined a Hidden Markov Model (HMM) over a set of hidden user states. The observable user purchase activity in a time interval is dependent to the users inner state in the same time interval. We picked

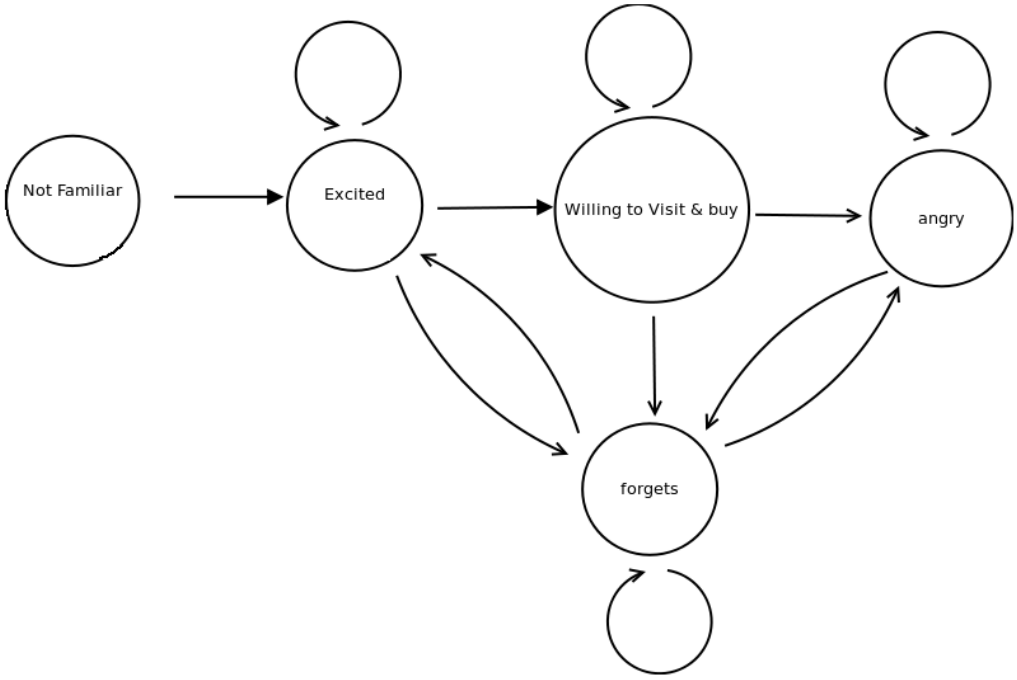

Figure 3–1: Structure of the customer HMM

the structure of the model, but the observation parameters have to be learned from the available data.

We built a simple model with 2 observations (purchase or not) and 5 internal states:

1. Not familiar
2. Excited
3. Willing to visit and buy
4. Angry
5. Has forgotten about the web site

The structure of the model is depicted in Figure 3–1.

The initial probability vector is $\pi_0 = [1, 0, 0, 0, 0]$ since the user always starts in a"not familiar" state. We chose an initial probability matrix intuitively based on how likely transitions might be:

$$A_0 = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0.65 & 0.2 & 0 & 0.15 \\ 0 & 0 & 0.45 & 0.15 & 0.4 \\ 0 & 0 & 0 & 0.8 & 0.2 \\ 0 & 0.2 & 0 & 0.2 & 0.6 \end{bmatrix} \tag{3.1}$$

These parameters can be modified using the EM algorithm, if one wants to improve on the structure. But we decided to use a fixed initial setting in order to constrain the model to a structure which seems reasonable.

The most important part of the model from our point of view is the observation model, because intuitively it should take into account not only the HMM internal states, but also the context of the events that are happening. Because of this, we experimented with two different models. One is the classic HMM, in which only the customer state matters, and the observations are binomial distributions dependent on the state.

The other model we tried is to model the observation probabilities using logistic regression. In this case, for every internal state of the users, we will compute one parameter vector for a logistic regression. The input to the logistic regression is given by an event description. The output is interpreted as the probability of the purchase. Hence, the logistic regression predicts the purchase probability given an internal state of a user, and a set of event features.

The reason we chose logistic regression for this step is that we want to make sure that we can condition on the 100-dimensional continuous vector representing events. This is fairly large, so if we chose a more complicated model we might have very high variance in the learning process. At the same time, the output of the logistic regression has the desired probabilistic meaning.

We will fit the logistic regression parameters based on the data from all users. But because each model depends on the internal state, for each user we will need to estimate their state before we decide to which of the 5 models we should give the data. Because multiple events are going on at the same time, we created multiple instances for each time step, corresponding to each event.

We will refer to the model that uses the HMM and the logistic regression together as the hybrid model.

3.3 Learning algorithms

The pure HMM model will be trained by the standard Baum-Welch algorithm as described in Chapter 2. Hence, we will focus on the hybrid model, which requires an adapted training algorithm. If the state s_t of the user is given, and we have a set of event vectors $I_t = \{E_1^t, \dots E_k^t\}$, we compute the probability of a purchase by creating inputs corresponding to all the events. We then compute the outputs for each of $E_1^t, \dots E_k^t$ by using the weight vector w_{s_t} corresponding to the state. We then average the probabilities obtained to give a global answer. The state s_t is obtained by using forward inference and sampling the user's current state based on the past data. We avoided using the Viterbi algorithm because in a realistic setting, we want

to infer the purchase of the user in the future, so forward inference is the natural approach.

Note that we could have also chosen to use the belief distribution over s and average the answers over all internals states with the appropriate probabilities. The reason we did not use this approach is that we want to avoid too much averaging, because it drives probabilities towards 0.5, which is pure chance.

In order to train the model, we start with 5 logistic regression vectors w_s each of length 101 (to accommodate the 100 events variables and a bias term equal to 1). The initial weights are drawn uniformly between $[-1, 1]$. This interval is picked because of the fact that LR generally performs better with the initial parameters closer to zero.

We fix the transition and initial probabilities to the values described above; these will not be learned by the algorithm. Instead, the algorithm focuses on updating the parameters w. The training algorithm loops over all users u, and for each user does the following (dropping u for clarity:

1. q_1 is drawn from π (in our case, this is deterministic since we always start in the same state)
2. For each time step $t = 1$ to T:
   (a) Construct a new data set for the logistic regression, by creating examples with all the corresponding events in I_t as input and O_t as the output, and adding it to the data set for q_t

(b) Compute the new state:

$$Q_{t+1} \sim \frac{P(Q_{t+1}, O_{t+1}|O_1, \ldots O_t, A, w)}{P(O_{t+1}|O_1, \ldots O_t)} = \frac{\sum_{j \in S} P(Q_t|O_1, \ldots O_t) a_{ji} p(O_{t+1}|\phi_{t+1}, w_j)}{\sum_{i \in S} \sum_{j \in S} P(Q_t|O_1, \ldots O_t) a_{ji} P(O_{t+1}|\phi_{t+1}, w_j)}$$

(c) $t \leftarrow t + 1$

3. Update all the weights w_i using logistic regression on the new data set. We use the Newton-Raphson method, which does not have a learning rate.

We note that different solutions are possible in the last step, in terms of whether one should gather the data set over all users, or just subsets of users, or indeed not use Newton-Raphson and instead just update weights incrementally after seeing each user.

CHAPTER 4
Experiments and results

We now describe the experimental setup and the results using the algorithmic approach and data discussed in Chapter 3.

4.1 Experimental setup

We used the Matlab machine learning toolbox developed and maintained by Kevin Murphy since 1998, as a basis for both the hidden Markov model and the logistic regression algorithms, and available at: http://www.cs.ubc.ca/ murphyk/Software/. This toolbox supports inference and learning for HMMs with discrete outputs (dhmm), Gaussian outputs (ghmm), or mixture of Gaussians output (mhmm). The inference routine supports filtering, smoothing and fixed-lag smoothing of the data. For our problem, we used this dhmm part of the toolbox. We programmed the HMM-LR by building on this code. The program is fully compatible with the toolbox, which will be useful if others want to use our approach, as this toolbox has been used extensively in the machine learning and probabilistic modelling communities.

In all experiments, in order to evaluate fairly the performance of the algorithms, we used k-fold cross validation. In k-fold cross-validation, the data is split into k equal partitions, trying to ensure an even distribution of data in each partition. As we deal with time series, the trajectories have to be split in such a way that each trajectory is exactly in one fold. Then, each fold is picked in turn to be the testing fold. The data in the other $k-1$ folds is used for two purposes. Part of the data is

used to train the parameters of the model. The other part is used as a validation set; it allows evaluating the models more objectively, so one can decide which parameters (such as initial choices, learning rates, model structure) are best. In the experiments, we used $k = 5$, which means that we split the users into 5 equal sets. We always used 3 sets for training, 1 set for validation, and 1 set for testing. The same data splits were used for all algorithms.

To evaluate the performance of the algorithms, we measured both the accuracy of the prediction and the average log-likelihood of the data. The accuracy measures how well our predictions of purchasing behavior match the observed actions of the users. The log-likelihood is a measure of the quality of the model, in terms of how well it predicts the observed data. We now describe how these measures are computed.

The average log likelihood of the models is the probability of the data given these models. For this, we use inference to compute the probability of the data (training or testing) after each EM iteration. We then take the log of this probability and average it over the examples. This corresponds to a geometric averaging of the actual probabilities. We expect the log likelihood of the model to increase over EM iterations, and this is what we plot in the results. For all models, log-likelihoods were used rather than direct probabilities in order to get better numeric accuracy and also for computational simplicity.

To measure accuracy, we compute the predicted probabilities of a purchase for each user at each time step, conditioned on the trajectory so far. Let $O_t(u)$ be the estimated probability of purchase for the user u at time step t. If this probability is above a threshold, we consider the user will make a purchase. In the experiments,

we use a threshold of 0.7 for this decision. Similarly, if $O_t(u)$ is below a threshold, we are fairly certain the user will not make a purchase, and we make this prediction. In the results we present, we use a threshold of 0.3. All predictions between these threshold are considered invalid, as the uncertainty is very high. Since we know the real observations on the data on which we evaluate, we can measure accuracy as the percentage of true predictions made over all users and all time steps (i.e. predictions that agree with the reality) out of the total number of predictions made (both valid and invalid).

In all experiments, all implementations ran 3 times, starting with different initial observation model parameters. The resulting of accuracy and log likelihood are averaged on these 3 independent runs. Ideally, more runs should be done, but time limitations prevented more runs at this point.

Note that the training algorithms are designed to maximize the likelihood of the data, but accuracy is a better performance measure in order to reflect the success of the models.

4.2 Experiments on simulated data

In order to validate first our HMM-LR proposed model, we tested it on data obtained by simulating a model of this type in which we know both the underlying user model and the internal user state. As we discussed in Chapter 3, we are using one logistic regression model for each state and since we do not know the real states, we guess them using the Viterbi algorithm. By using sample data with known states we want to check that the proposed model and learning algorithm work in an ideal situation.

We generated user purchase time series based on a specific hybrid model, keeping track of real state sequences, and applied those known state sequences to learn logistic regression model parameters. First, we describe the data generation process and then we describe the results.

4.2.1 Obtaining simulated data

Self-generated data for our primary experiments includes sample user purchase observation sequences and sample events. For simplicity we assumed events to have just one feature and each time step has just one event. Hence, in this data, E_t is the single value of the event taking place at time step $t \in \{1, 2, \ldots T\}$, whose value is drawn from a uniform distribution over $[0, 1]$. We chose $T = 10$.

We set π and A for the HMM-LR model to the model given in Equation 3.1 and picking a weight vector w uniformly from $[0, 1]$ for each state. We then generated $NU = 100$ time series of "fictitious" users, each of length T. To do this, we needed to generate the state of user u in each time step first and then generate the purchase observation in the same time step using the state.

The initial state probabilities are determined as:

$$p(Q_1(u) = j) = \pi_j \tag{4.1}$$

Then, at every successive time step, we compute the next probability distribution as:

$$p(Q_t(u) = j | O_1(u), \ldots O_{t-1}(u)) = \sum_{i \in S} P(Q_{t-1}(u) = j | O_1(u), \ldots O_{t-1}(u)) a_{ij}$$

where $P(Q_{t-1}(u) = j|O_1(u), \ldots O_{t-1}(u))$ is computed as in Equation 2.3. After generating $Q_t(u)$, we sample from it to obtain an actual state, j. Now, given j, we compute the output of the logistic regression for E_t as input and using weight vector w_j. This gives us the probability of a purchase event, $P(O_t(u) = 1)$. We now sample $O_t(u)$ with this probability.

All generated trajectories, including the values of the hidden states and the observations, are saved for future use.

4.2.2 Results

In these experiments, we use the learning algorithm described in Chapter 3 to learn the parameters of the logistic regression models.

In the first set of experiments, we wanted to sanity-check the learning algorithm. Hence, since we know the hidden state identity, we can use the exact identity of the state at every time step. We update the parameters of the logistic regression after every h users, and we experimented with $h = 1, 10$.

The results for this setup are presented in Figure 4–1 (log likelihood of testing data) and Figure 4–2 (accuracy). As expected, the algorithm quickly and accurately learns the correct model.

In a second experiment, we used the same simulated data, but instead of using the known identity of the state, we used the learning algorithm described in Chapter 3, which infers the identity of the state. We ran the algorithm for 10 iterations. The log-likelihood as a function of the amount of training is given in Figure 4–3 and the accuracy in Figure **??**. As can be seen from the figures, the performance drops significantly in this case, because the data used to learn the logistic parameters

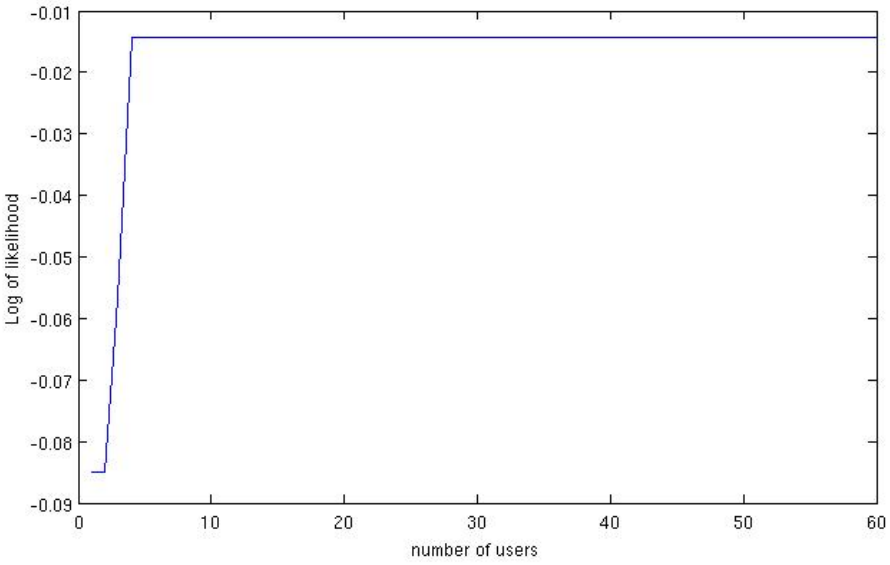

Figure 4–1: Average log likelihood measured on the test data using known hidden states

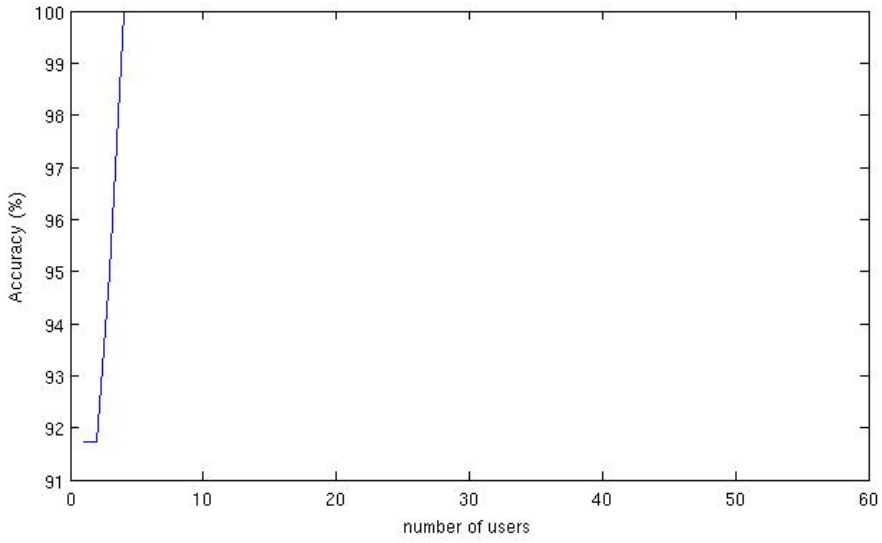

Figure 4–2: Average accuracy measured on the test data using known hidden states

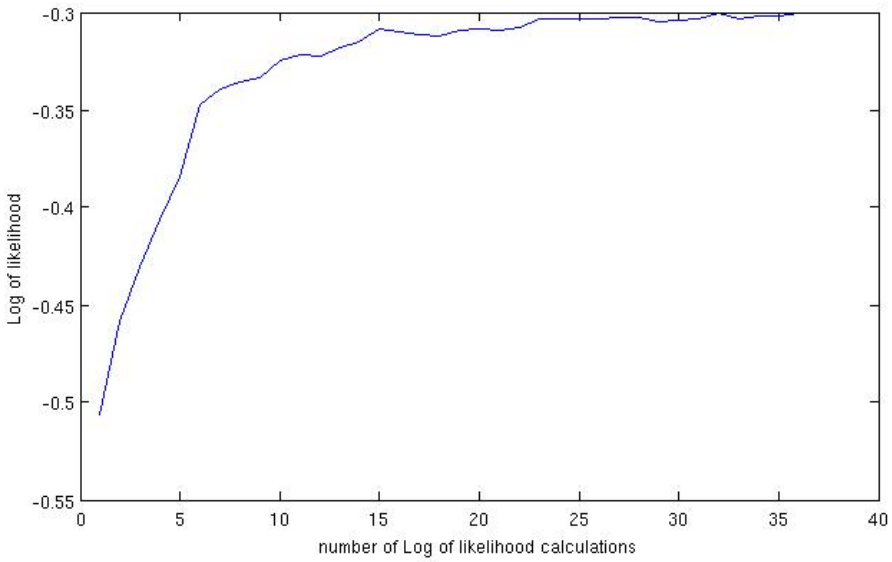

Figure 4–3: Average log likelihood measured on the test data using known hidden states

becomes a lot more variable. The accuracy achieved drops to roughly 65% at the end of learning. This suggests that some more investigation will be needed in the future to figure out better ways to infer the hidden states.

To see if the proposed algorithm provides a benefit compared to a simple HMM, we used also the HMM architecture with usual binomial observations and the model presented in Chapter 3 and learned its parameters using EM. The results are presented in Figures 4–5 and 4–6. As can be seen, accuracy here is a lot lower, attaining only about 50%, so using the logistic model helps.

We now turn to the real data. Here, we do not have events on every time step, so if there are no events, we use the usual HMM binomial model. For events, we use the logistic model as before. The log-likelihood and accuracy for the hybrid algorithm

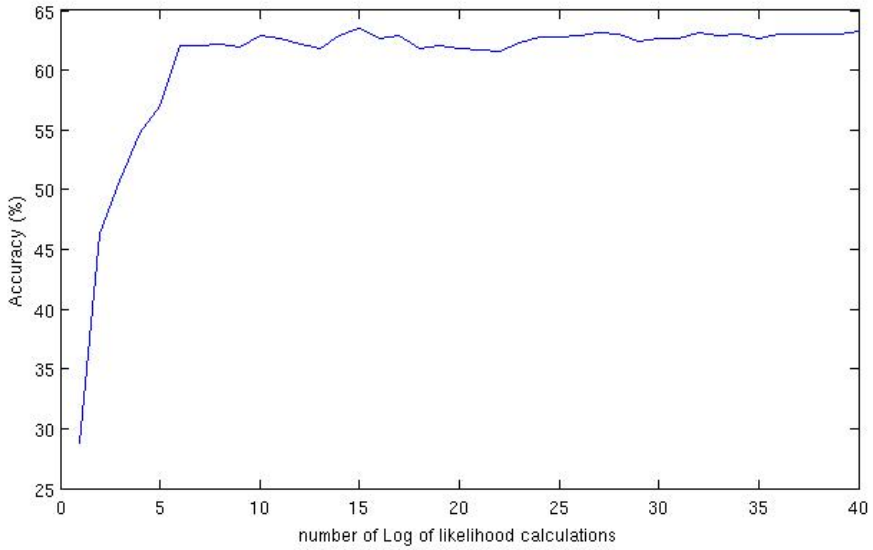

Figure 4–4: Average accuracy measured on the test data using known hidden states

are presented in Figure 4–7 and 4–8. As can be see, the achieved accuracy is lower. Note that the real data set is much larger than the simulated data set, due to which we had to limit the number of iterations. Similar results for the traditional HMM are in Figures 4–9 and 4–10.

After the last iteration, in sample data the likelihood and accuracy of the test data for hybrid model is 0.6552 (-0.4228Log of likelihood) and 62%, and the likelihood and Accuracy of the test data for HMM model is 0.6310 (-0.4603 Log of likelihood) and 81%. Regarding the result figures, the HMM-LR hybrid model works better than the pure HMM both for sample data and real data except for the Accuracy of real data. It seems pure HMM model predicts with higher probability both for right and wrong predictions. Obviously, hybrid model works very well when the paths and states are known for the sample data. As expected, the hybrid model with real data

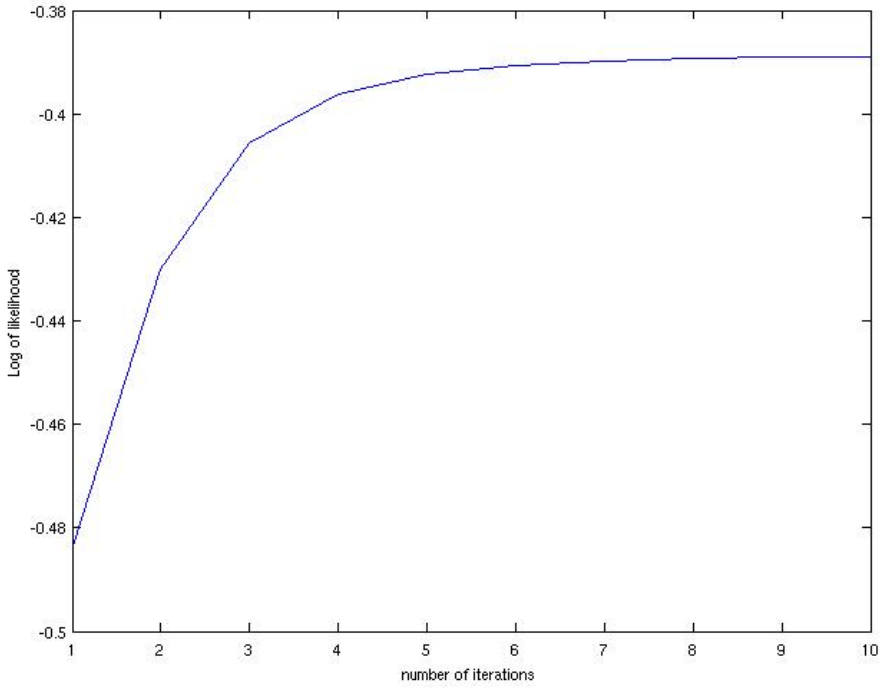

Figure 4–5: Average log likelihood measured on the test data using known hidden states

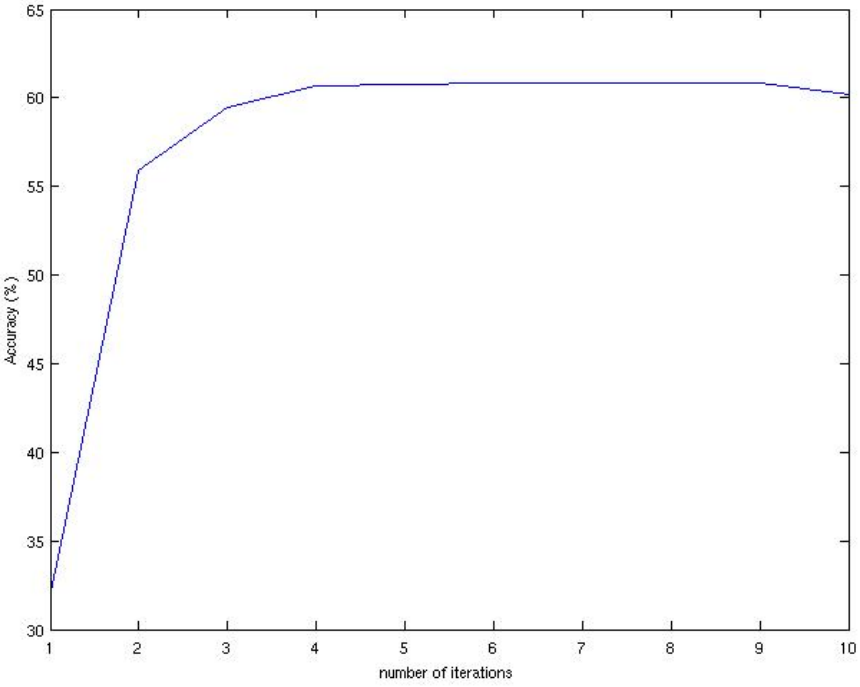

Figure 4–6: Average accuracy measured on the test data using known hidden states

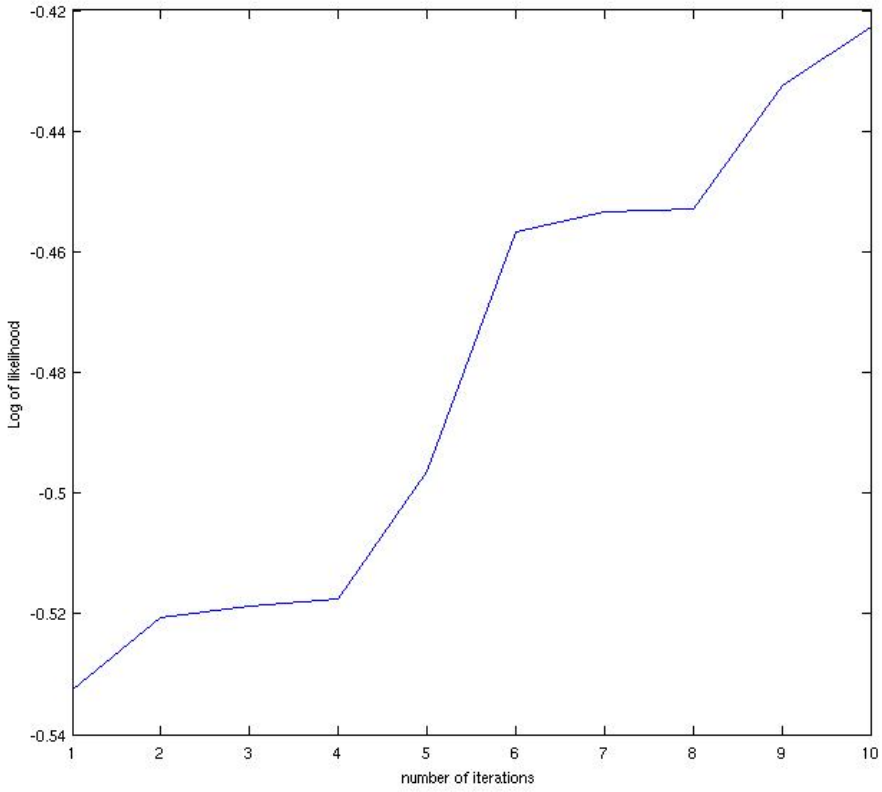

Figure 4–7: Average log likelihood measured on the test data using using real data for proposed model

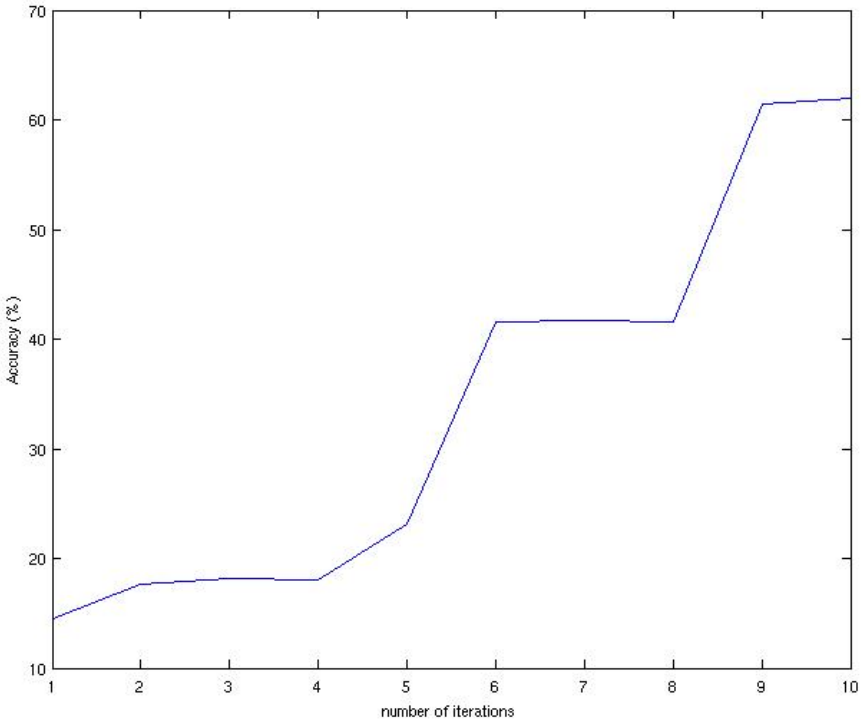

Figure 4–8: Average accuracy measured on the test data using real data for proposed model

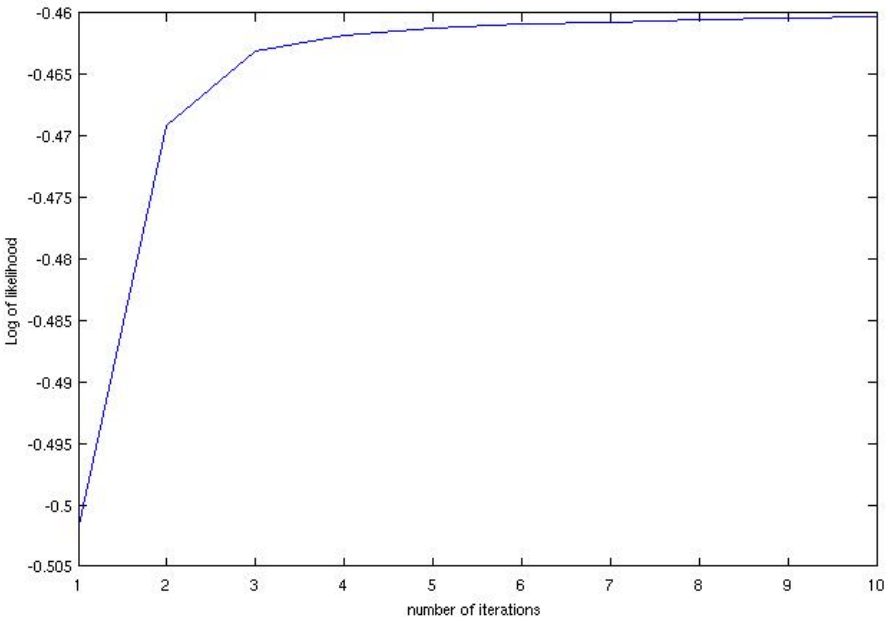

Figure 4–9: Average log likelihood measured on the test data using real data for traditional HMM

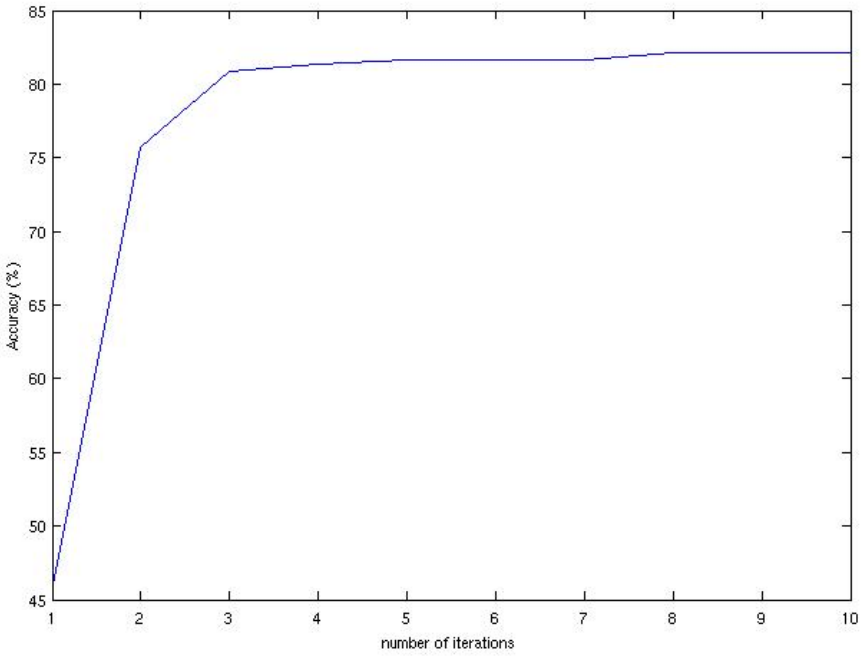

Figure 4–10: Average accuracy measured on the test data using real data for traditional HMM

is not working as great as the hybrid model with sample data and unknown paths, due to many other factors affecting the purchase output in the real data, but it still works better than pure HMM in terms of Log of likelihood. In sample data, after the last iteration, the likelihood of the test data for hybrid model and unknown paths outperforms the likelihood of the HMM model by 6%, and for known states and paths it is 31%. For real data, this amount is 2.5% and the hybrid model is slightly better than HMM. Learning converges after 10 iterations for the usual HMM model but for the hybrid model the log-likelihood and accuracy are still improving at this point. Due to the computational time complexity, we only did 10 iterations, but in the future we plan to make the code more efficient, so we can run these experiments longer.

CHAPTER 5
Conclusions and Future Work

We tried a standard HMM model and our proposed hybrid model of HMM-LR for our problem of predicting online user behavior toward ads. The latter approach is taken to support mutual influence of inner state of the user and advertisement properties on the purchase behavior of the user. For our simulated data, the hybrid model outperforms the HMM model, especially with known state paths. For the real purchase and advertisement event historical data, HMM-LR works slightly better than the HMM. The results support our the HMM-LR approach, but they are not yet of a quality that would allow this model to work in a real, on-line setting.

One issue which affects the performance of the HMM-LR model is the time inconsistency between the purchase data (members and orders data) and the events data. The first one was two years long but the second one was only six months long. If more data on advertising events were available, we anticipate that the results would be better. One other challenge in this problem is that if the real probabilities of purchasing and not purchasing are close to each other, it is difficult for an approximate probabilistic model to make the right prediction. On our simulated data, the model works much better when we simulate data with very distinct distributions for purchase and not purchase outputs, than if these probabilities are close. Moreover, in our data there are usually several events overlapping in each time step and we associate the user behavior to all of them. However it is more likely in reality that

the purchase observation of the user is influenced by just one of these ads. The current training setup treats this problem as noise, but if one could make a good guess about which ad had most influence, results would likely be much better. In our data set, we only had access to feature vectors describing events and purchased products. Using the actual text (to which we did not have access) would allow a better association of a purchase with one particular ad.

In addition, we have a very simple state set in our model. We made this choice because we wanted to make sure that the model can be trained well from the data we had. However, the simplicity in the model likely introduces bias in the results. Probably considering a more complex state set would yield in a better result.

One of the main issues is that there are many more factors influencing the purchase behavior of the real users, like changes in their financial state and special holidays. In the future work, these factors could be considered. Furthermore, in the problem, the overall number of purchases is much lower than not purchases and this makes the learning even more difficult. Methods for sampling the purchase and non-purchase events to improve the problem of class imbalance should be considered as future work.

The type of model we proposed can be used as a part of an online system for ad customization for users and also for predicting ad value and for selecting ads based on the average predicted user behavior. In this setup, user response could be used as new data to train the model online. The extended problem is to find the marginal value of ads and instead of having a binary observation of purchase and not purchase, having more classes for different purchase ranges or even considering a continuous

value output predicting for the amount of purchase. This problem could be tackled easily with our current model by first predicting the purchase or not purchase class and then predicting the purchase value by regression.

Overall, our experiments confirm the utility of machine learning in mining large amounts of online customer data. We proposed one particular type of probabilistic model and explored its effectiveness, with encouraging results in a large real data set. The improvement directions discussed above could leave to even better results in the future.

References

[1] Ajith Abraham and Ramos Ramos. Web usage mining using artificial ant colony clustering and linear genetic programming. In *The 2003 Congress on Evolutionary Computation (CEC '03)*, pages 1384–1391, 2003.

[2] Dimitris J. Bertsimas, Adam J. Mersereau, and Nitin R. Patel. Dynamic classification of online customers. In *Proceedings of the Third SIAM International Conference on Data Mining (SDM'03*, 2003.

[3] Bruce DAmbrosio, Eric Altendorf, and Jane Jorgensen. Relational Bayesian models of on-line user behavior. In *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge discovery and data mining (KDD'03)*, 2003.

[4] Xin Jin, Zhou Yanzan, and Bamshad Mobasher. Web usage mining based on probabilistic latent semantic analysis. In *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge discovery and data mining (KDD'04)*, pages 197–205, 2004.

[5] Chun-Jung Lin, Fan Wu, and I-Han Chiu. Using Hidden Markov Model to predict the surfing users intention of cyber purchase on the web. *Journal of Global Business Management*, 2009.

[6] Geoffrey J. McLachlan and Thriyambakam Krishnan. *The EM algorithm and extensions*. Wiley-Interscience, 2007.

[7] Chihiro Ono, Mori Kurokawa, Yoichi Motomura, and Hideki Asoh. A context-aware movie preference model using a Bayesian network for recommendation and promotion. In *Proceedings of the 11th International Conference on User Modeling (UM'07)*, pages 247–257, 2007.

[8] Lawrence Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proc. IEEE*, 77(2)):257–286, 1989.

[9] Paul Resnick, Neophytos Iacovou, Mitesh Suchak, Peter Bergstrom, and John Riedl. Grouplens: An open architecture for collaborative filtering of netnews. In *Proceedings of ACM 1994 Conference on Computer Supported Cooperative Work*, pages 175–186, 1994.

[10] Nachiketa Sahoo, Param Vir Singh, and Tridas Mukhopadhyay. A hidden markov model for collaborative filtering. *MIS Quarterly*, 36(4):1329–1356, 2012.

[11] Neelam Sain and Sitendra Tamarkar. Web usage mining and pre-fetching based on hidden markov model and fuzzy clustering. *(IJCSIT) International Journal of Computer Science and Information Technologies*, 3(4):4874–4877, 2012.

[12] Weihua Song, Vir V. Phoha, and Xin Xu. The HMM-based model for evaluating recommender's reputation. In *IEEE International Conference on E-Commerce Technology for Dynamic E- Business*, pages 209– 215, 2004.

[13] Katsutoshiv Yada and Natsuki Sano. Customer behavior modelling using radio frequency identification data and the Hidden Markov Model. *Service Research and Innovation Institute Global Conference*, pages 509–514, 2012.