Mobile Health Monitoring: The Glucose Intelligence Solution

Chun Ming Chin

Electrical Engineering and Computer Sciences
University of California at Berkeley

Technical Report No. UCB/EECS-2012-160
http://www.eecs.berkeley.edu/Pubs/TechRpts/2012/EECS-2012-160.html

June 3, 2012

Acknowledgement

Donald Jones, Vice President of Qualcomm Wireless Health at San Diego, California.
Paul Nerger, Chief Technical Officer of a Silicon Valley mobile healthcare start up, Happtique (www.happtique.com).
Alexander Smola, machine learning professor.

Mobile Health Monitoring: The Glucose Intelligence Solution

Chun Ming Chin

May 8, 2012

1 Abstract

Diabetes is a disease that causes a person's sugar levels to vary too high or too low. This has implications such as blindness, disability, and even death. Its cost is becoming a healthcare burden for the United States, as more people are afflicted with diabetes. We build mobile healthcare applications to help diabetics better monitor their condition, save healthcare cost and improve the quality of life. Among the solutions we investigated was a prediction algorithm that estimates how a person's glucose levels will change based on specific food and exercise inputs. Another solution was a recommendation system that suggests diabetic-friendly food options when a person shops for groceries or eats at a restaurant. Our solutions are tested on ten people with diabetes over three months. Based on our tests, users found our recommendation system which suggests diabetes-friendly restaurants helpful in controlling their condition when eating out.

2 Introduction

Diabetes is a costly chronic disease that affects about 10% of the population in the United States [1]. This problem is not only widespread, but is also growing. It is expected that by 2020, one out of three Americans would be diabetic, and medical cost runs up to $132 billion per year in the United States [2]. To address this problem, we are building a personalized mobile healthcare application that lets diabetic patients better monitor their glucose levels so that they avoid costly consequences such as blindness, amputation and even death.

This is an interesting problem to solve as mobile devices help shift healthcare away from inconvenient, expensive central locations. Individuals can now monitor their health in their daily lives. This is especially helpful for patients such as the aged and people with physical disabilities. In particular, this saves time because people no longer have to wait at hospitals to be attended by a limited number of healthcare staff - mobile healthcare monitoring removes this administrative bottleneck. In addition, mobile healthcare monitoring is more cost effective as it enables preventive monitoring rather than costly treatment. In the long run, by solving the widespread diabetes problem, we can port our solution to other chronic diseases such as Leukemia and Anemia.

This paper first explores literature that describes algorithms which are relevant to the problem we want to solve. These problems are discovered from interviews and surveys we have conducted on people with diabetes in the bay area. Next, to solve such problems, we describe the solutions we have built using the mobile device. This includes a prediction algorithm that informs a patient on how to control their glucose levels through exercise and proper nutrition. Such advice is founded upon the historical data of the patients lifestyle. We also built a recommendation system that helps people with diabetes eat out smartly. Finally, we report the feedback we collected from our recommendation and prediction algorithms.

3 Literature Review

3.1 Measuring Blood Glucose Variability

This paper is relevant to our project problem of predicting glucose level readings as it models variations in Blood Glucose Level (BGL) using a stochastic model [3]. Each glucose reading is modeled as a Gaussian stochastic process with time-varying mean and variance. This probability distribution is sampled using an algorithm known as the Monte Carlo Markov Chain method. The best estimate for each glucose reading is then determined by computing the mean and standard deviation of each point. The point estimates have small mean squared error when it is used to fit synthesized data, and works reasonably well on blood glucose measurements taken from actual patients.

We can use the this algorithm for two purposes on our project:

1. Use the stochastic model as a foundation to build a prediction algorithm.
2. Port algorithm onto a mobile device to help diabetic patients ensure that their blood glucose levels are within safe bounds.

However, there are shortcomings to the author's approach. The Monte Carlo Markov Chain algorithm is computationally expensive. This problem is amplified further since the algorithm has to be run on a resource-constrained mobile device. Moreover, simpler methods such as moving average and regression can achieve similar accuracies with less expensive computation.

3.2 Predicting Glucose Levels With Continuous Glucose Data

This paper implements a prediction algorithm based on continuous glucose monitoring (CGM) time series data. [4] Two methods are used for prediction.The first uses a first-order polynomial, while the second one uses a first-order auto regressive model. The results from both models demonstrate that the algorithm can be used to ensure a patient's blood glucose levels is always within safe bounds (i.e. Prevent hypo and hyper glycemic events.). However, the paper's short coming is that it depends on data drawn from a continuous glucose measurement hardware device. This is typically a bulky, expensive product that has to be attached to a diabetic person's body. Such hardware is not used by the majority of people with diabetes. Diabetics often use test strips, which meant data can only be gathered about five times in a day rather than continuously. Therefore, this prediction algorithm is not an attractive solution.

3.3 Recommendation System

For our recommendation system to enable people with diabetes to eat out smartly, we base our implementation on the "Collaborative Filtering with Temporal Dynamics" algorithm by Yehuda Koren from Yahoo! Research [5]. This paper was used to improve the accuracy of Netflix's movie recommendations.

The paper uses the collaborative filtering to compare two fundamentally different objects: items against users. There are two approaches to accomplish this: the neighborhood approach or latent factor models.

Neighborhood methods compute the relationship between items or between users. For example, an item-item approach evaluates the preference of a user to an item based on ratings of similar items by the same user. A user-user approach evaluates preferences of a user to an item by referencing other users who share the same ratings as those that the original user rated. In other words, these methods transform users to the item space in order to make relevant comparisons. On the other hand, latent factor models transform both items and users to the same space, known as a latent factor space for comparison. For example, when items are movies, factors measure variables such as comedy versus drama or the amount of action, which users can relate to.

For the literature review, an investigation into the theoretical foundations of the neighborhood method is conducted. Later, in the methodology section, I implement a prototype of the latent factor model, known specifically as Matrix Factorization.

3.3.1 Neighborhood Method

Neighborhood methods are derived from relationships between items or users. In the first case, the algorithm recommends items that also have similar ratings given by other users. This is known as the item-item similarity model. The concept is framed by the statement "Users who like this restaurant also like this restaurant". In the second case, the algorithm recommends items that other users also like. This is known as the user-user similarity model. This is good if the item base is smaller than the user database.

This paper is useful as a reference for the recommendation system but it's mathematical approach is not suited for our use case to recommend restaurants. In particular, while the author uses an item-item based model for its neighborhood method, we believe it would be more relevant if we use a user-user model instead for our diabetes context. This means that we should recommend restaurants and food options based on user-user similarities such as demographic, cultural and physiological factors.

An in-depth literature review into the mathematical motivations of the neighborhood method is conducted [6] because it informs the implementation of our recommendation system later. The rating system for the movie rating metric is represented graphically in Table 1.

Table 1: Neighborhood based Collaborative Filtering. Blank spaces denote unknown data. Numbers denote rating between 1 to 5

	Users				
Items	1	2	3	4	5
1	1		3		
2			5	4	
3	2	4		1	2
4		2	4		5
5			4	3	4
6	1		3		3

A simple way of determining missing values in Table 1 is by using the concept of weighted average. Lets say we want to estimate the rating of item 1 given by user 5. We find that items 3 and 6 are similar to item 1 because user 1 rates them similarly. Assuming the measure of similarity between items i and j is represented by the variable s_{ij}, we can estimate the weighted average score of the rating for item 1 by user 5 with the equation:

$$r_{15} = \frac{2 * s_{13} + 3 * s_{16}}{s_{13} + s_{16}} \tag{1}$$

Where s_{13} is the similarity measure between item 1 and 3 while s_{16} is the similarity measure between 1 and 6. For example, if $s_{13} = 0.2$, $s_{16} = 0.3$, the weighted average is then: $\frac{0.2*2+0.3*3}{0.2+0.3} = 2.6$.

The similarity measure s_{ij} between 2 items, i and j can be determined using the correlation

coefficient:

$$s_{ij} = \frac{Cov[r_{ui}, r_{uj}]}{Std[r_{ui}]Std[r_{uj}]} \tag{2}$$

Where $Std[r_{ui}]$ and $Std[r_{uj}]$ is the standard deviation of the known user ratings for item i and j respectively. This is shown graphically in Table 2. We only take values for item i and j where both exists.

Table 2: Computation of item-item similarity

User ratings for item i					
i	1	?	5	3	?
User ratings for item j					
j	?	1	2	5	?

However, this naive implementation shown in equation (1) fails to address the following methodological issues:

- We need an algorithm that scales well with increasing number of users and items. In implementation of equation (1), there is a lot of lookups, which slows down the performance of the algorithm.
- Some restaurants may have a lot of reviews, while others do not have any reviews. Such an imbalanced dataset may cause unreliable recommendations
- When a new user registers at the Diabeats website, there is the challenge of what should you recommend to him when he did not make any reviews yet. This is an issue know as the cold start problem. It is important to address this because if a user does not see relevant content at the start. They will leave the Diabeats site.
- Some users rate substantially stricter than others. Our recommendation system should be able to correct such user rating biases.
- Ratings change overtime

In order to address these issues, we used a different approach to fill the blanks in Table 1 with our own baseline estimation for a particular user and item, we use the following equation:

$$b_{ui} = \mu + b_u + b_i \tag{3}$$

Where b_{ui} is the baseline estimation of restaurant i by user u. μ is the average rating of all restaurants (i.e. the common offset). b_u is the average opinion of the user with restaurants in general (How strict the reviewer is), and b_i is how good the restaurant is inherently.

We want to minimize the difference between our baseline estimation b_{ui} and the actual rating r_{ui} of the restaurant given by the user. This can be expressed as a least mean squares problem:

$$\begin{aligned} & minimize_b \sum_{u,i} (r_{ui} - b_{ui})^2 + \lambda[\sum_u b_u^2 + \sum_i b_i^2] \\ = & minimize_b \sum_{u,i} (r_{ui} - \mu - b_u - b_i)^2 + \lambda[\sum_u b_u^2 + \sum_i b_i^2] \end{aligned} \tag{4}$$

We sum over all the pairs of user u and restaurant i where the user actually rated something. The expression $\lambda[\sum_u b_u^2 + \sum_i b_i^2]$ is used to penalize the coefficients b_u and b_i so that the user and restaurant biases becomes zero. The higher the value of λ, the smaller the coefficients for b_u and b_i and the more you will just take the mean μ

We can solve the least mean squares problem to find b_u and b_i the following way: Firstly, to find b_i, differentiate equation (2) with respect to b_i and set the result to zero:

$$\sum_{u \epsilon R(i)} -2(r_{ui} - \mu - b_u - b_i) + 2\lambda \sum_u b_i = 0 \tag{5}$$

Of the entire sum over pairs of u and i, the pairs that matter are those where the users actually went to restaurant i. Therefore, this is represented by the symbol $\sum_{u \epsilon R(i)}$, where R(i) is the set of items i. Rearranging the above equation to get b_i, we have:

$$[\lambda + |R(i)|] * b_i = \sum_{u \epsilon R(i)} (r_{ui} - \mu - b_u)$$

Hence

$$b_i = \frac{\sum_{u \epsilon R(i)} (r_{ui} - \mu - b_u)}{[\lambda + |R(i)|]} \tag{6}$$

Where $|R(i)|$ is the size of the set i.

In the same way, we get b_u to be:

$$b_u = \frac{\sum_{i \epsilon R(u)} (r_{ui} - \mu - b_i)}{[\lambda + |R(u)|]} \tag{7}$$

Where $|R(u)|$ is the size of the set u

By plugging the the solutions of b_i and b_u back to equation (3), we get good baseline estimates.

4 Methodology/ Approach

4.1 Diabetic Lifestyle Recommendation System

Our latest product iteration is a web application which empowers diabetics to eat out smartly. The website is at Diabeats.com (http://www.diabeats.co/). It is a community of diabetics sharing their knowledge on where are diabetic-friendly places to eat out at restaurants. You can search for food venues and read restaurant reviews that other diabetics with similar demographic profiles enjoy. At the core of Diabeats is a recommendation system that filters and ranks the restaurants around you based on your preference. I designed the backend recommendation system on Matlab while the rest of the team built the front end web use interface and back end server connection in HTML5 and Javascript

The recommendation system makes automatic predictions (i.e. filtering a list of restaurant results) about the interests of a user by collecting preferences or taste information from other users (i.e. collaborating). This method is known as collaborative filtering.

The assumption of the collaborative filtering approach is that if a person A has the same opinion as a person B on an issue, A is more likely to have B's opinion on a different issue x than to have the opinion on x of a person chosen randomly. In the diabetes context, if patient A shares the same ethnicity, gender and lifestyle choices as patient B, A's diabetic condition is like to be similar to B's condition. By seeking people with similar physiological and lifestyle profiles, a patient can learn how to keep their glucose levels within safe bounds.

4.1.1 Matrix Factorization Method

I implemented a collaborative filtering method called Matrix Factorization on Matlab (Source code attached in appendix). Since the dataset for diabetic friendly restaurants are not available at the time of writing, the recommendation system is implemented first in a separate context - rating movies since the movie rating dataset is available from the the MovieLens collection (`http://grouplens.org/node/12`), where a collection of 10 million ratings is used. The dataset is downloaded from `http://www.grouplens.org/sites/www.grouplens.org/external_files/data/ml-10m.zip`. The mathematical problem of rating movies is fundamentally similar to

Figure 1: Restaurant Recommendation Matlab Prototype

the problem of rating diabetic-friendly restaurants. The latent variables used for the movie context can be translated to the diabetic context (i.e. carbohydrate count, fat content, amount of fibre etc.)

The following model is used to estimate the rating a user gives to a particular movie:

$$f(u, m) = b_u + b_m + \langle v_u, v_m \rangle \tag{8}$$

Where b_u is the average opinion of the user with movies in general (How strict the reviewer is), and b_m is how good the movie is inherently.

The following loss function is used:

$$\sum_{(u,m)\epsilon Rate} \frac{1}{2}(b_u + b_m + \langle v_u, v_m \rangle - y_{um})^2 + \frac{\lambda}{2}[\sum_u ||v_u||^2 + b_u^2 + \sum_m ||v_m^2|| + b_m^2] \tag{9}$$

To solve minimize this loss function, the joint stochastic gradient descent equation is used across the ith row and jth column, where i and j = 1, 2, 3 ... 50. We learn the latent factors jointly from data streamed from disc. Perform the following at each iteration of i and j. At each iteration, update a user, movie pair then move on to the next one.

$$\begin{aligned}
\nabla Loss(v_{ui}) &= (b_u + b_m + \langle v_{ui}, v_{mj} \rangle - y_{um})(v_{mj}) + \lambda v_{ui} \\
\nabla Loss(v_{mj}) &= (b_u + b_m + \langle v_{ui}, v_{mj} \rangle - y_{um})(v_{ui}) + \lambda v_{mj} \\
\nabla Loss(b_u) &= (b_u + b_m + \langle v_{ui}, v_{mj} \rangle - y_{um}) + \lambda b_u \\
\nabla Loss(b_m) &= (b_u + b_m + \langle v_{ui}, v_{mj} \rangle - y_{um}) + \lambda b_m
\end{aligned} \tag{10}$$

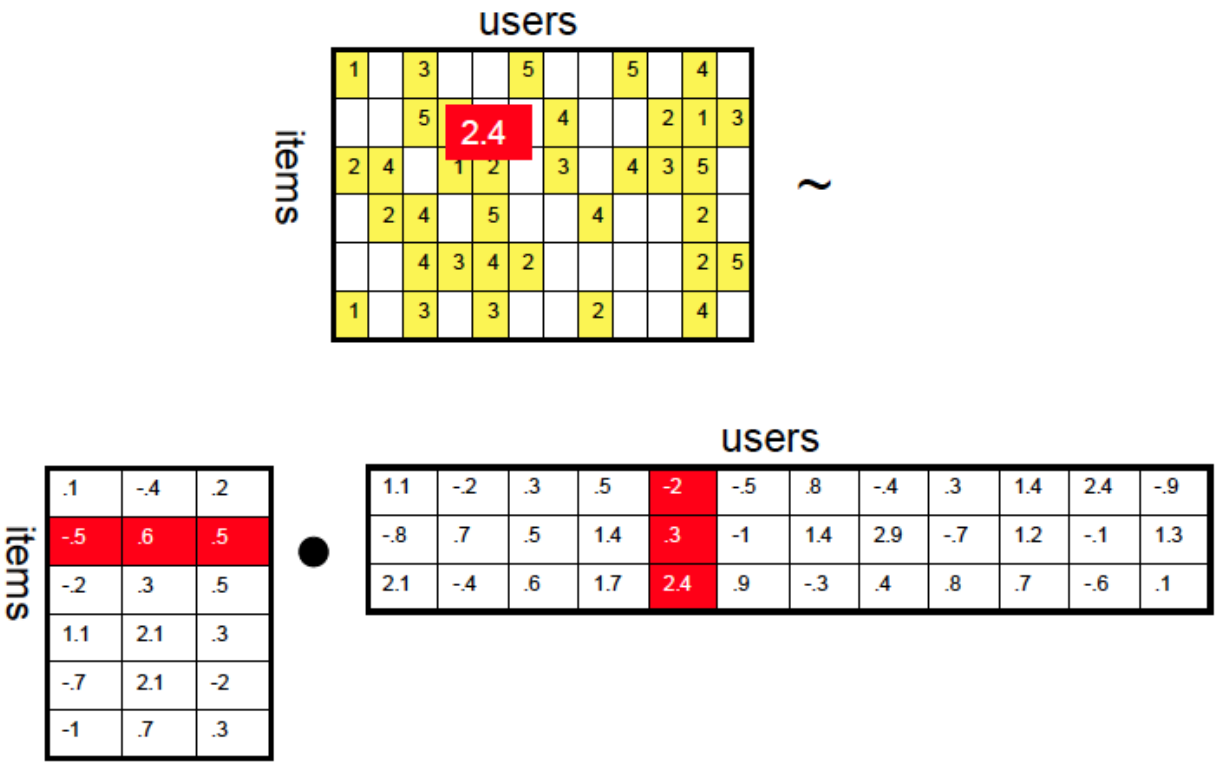


Figure 2: Estimate 5th user's unknown rating on 2nd item as inner products of latent factors that has 3 dimensions. Source: Scalable Machine Learning, Recommender Systems. Alex Smola. Yahoo! Research and ANU

In order to solve this convex optimization problem, take learning rate $\eta_t = \frac{1}{\sqrt{\alpha+\beta t}}$, where $\alpha = 1000$, which decides where you start and $\beta = 1000$, which decides how much you scale.

$$\begin{aligned} v_{ui(t+1)} &\leftarrow v_{ui(t)} - \eta_{(t)} \nabla Loss(v_{ui(t)}) \\ v_{mj(t+1)} &\leftarrow v_{mj(t)} - \eta_{(t)} \nabla Loss(v_{mj(t)}) \\ b_{u(t+1)} &\leftarrow b_{u(t)} - \eta_{(t)} \nabla Loss(b_{u(t)}) \\ b_{u(t+1)} &\leftarrow b_{u(t)} - \eta_{(t)} \nabla Loss(b_{u(t)}) \end{aligned} \tag{11}$$

Once we have determined the latent factors v_u and v_m from our training set, we can then use this to predict the rating a user may give a movie by doing a matrix multiplication with the corresponding v_u row and v_m column. However, this algorithm is still inaccurate for new users or new movies that have few ratings, since there is no prior information to train the latent factors corresponding to these v_u and v_m factors. This is also known as the cold start problem. This issue is solved by using the inherent movie attributes to improve our recommendation system. Movie attributes refer to the way movies can be categorized into different genres, such as 'Action', 'Adventure', 'Animation', 'Western' etc. To achieve this, we replace the movie attribute vector v_m by one that has categories added:

$$v_m \leftarrow v_m + \sum_{c \epsilon categories(m)} v_c \tag{12}$$

Here, categories(m) represents the set of categories specific to movie m. The new objective function with v_m replaced by $v_m + \sum_{c \epsilon categories(m)} v_c$ becomes:

$$\begin{aligned} \sum_{(u,m)\epsilon Ratings} \frac{1}{2}(b_u + b_m + \langle v_u, v_m \rangle + \langle v_u, \sum_{c \epsilon cat(m)} v_c \rangle - y_{um})^2 \\ + \frac{\lambda}{2}[\sum_u ||v_u||^2 + b_u^2 + \sum_m ||v_m||^2 + b_m^2] \\ + \frac{\lambda}{2} \sum_m \sum_{c \epsilon cat(m)} ||v_c||^2 \end{aligned} \tag{13}$$

Where b_u and b_m are scalars. v_u is a 1x50 vector and v_m is a 50x1 vector, with u = 1, 2, 3 ... (Total No. of Users). m = 1, 2, 3... (Total No. of Movies). v_c is a 50x1 vector, where c = 1, 2 ... 18. $c \epsilon cat(m)$ refers to the subset of categories out of the 18 (i.e. 'Action', 'Adventure', 'Animation', ... 'Western')that describe a specific movie. Extend the joint stochastic gradient descent algorithm

with the v_c parameter. Let $p = b_u + b_m + \langle v_u, v_m \rangle + \langle v_u, \sum_{c \epsilon cat(m)} v_c \rangle - y_{um}$. The loss function is:

$$\begin{aligned} \nabla Loss(v_u) &= (p)(v_{mj} + \sum_{c \epsilon cat(m)} v_c) + \lambda v_u \\ \nabla Loss(v_m) &= (p)(v_u) + \lambda v_m \\ \nabla Loss(b_u) &= (p) + \lambda b_u \\ \nabla Loss(b_m) &= (p) + \lambda b_m \\ \nabla Loss(v_{c \epsilon cat(m)}) &= (p)(v_u) + \lambda \sum_{c \epsilon cat(m)} v_c \end{aligned} \tag{14}$$

The update equation is:

$$v_{c \epsilon cat(m)(t+1)} \leftarrow v_{c \epsilon cat(m)(t)} - \eta_{(t)} \nabla Loss(v_{c \epsilon cat(m)(t)}) \tag{15}$$

Refer to Matlab script for the stochastic gradient descent algorithm implementation. The outcome of this modification is that movies with the biggest changes are those that have few ratings. Movies with many ratings have little change using the new algorithm.

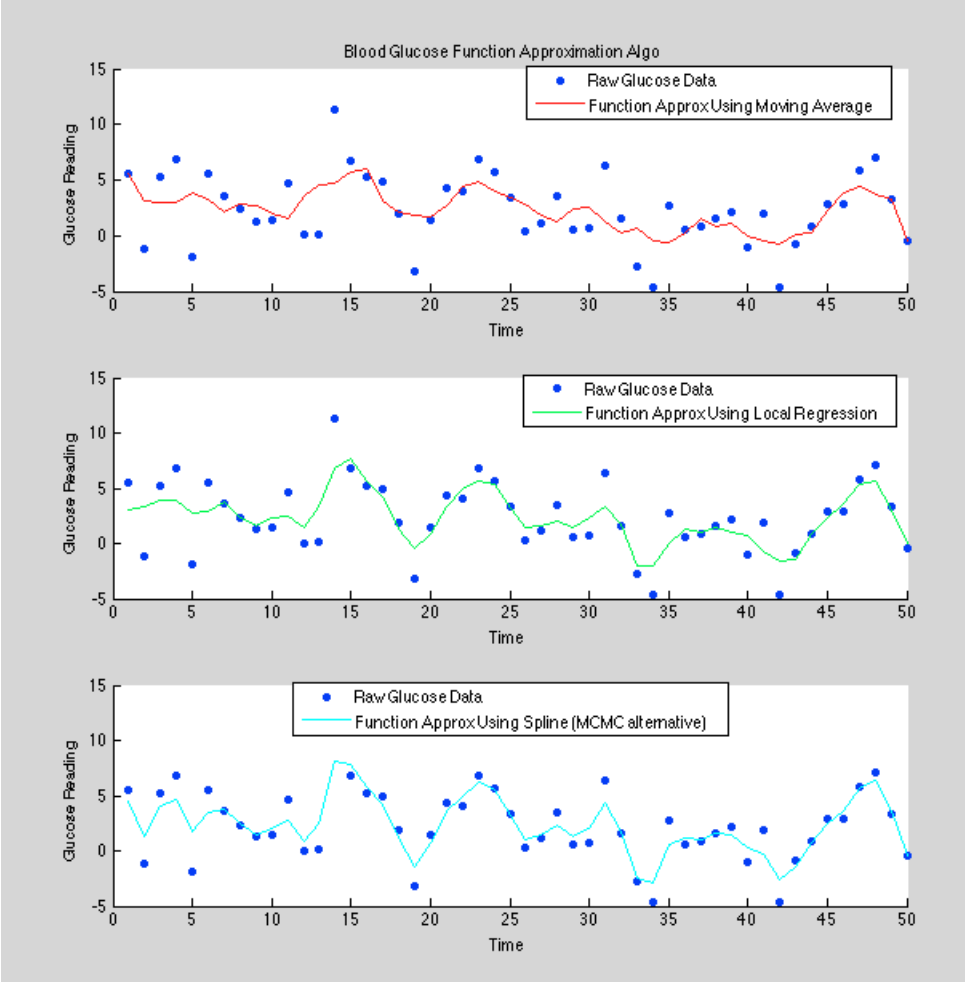
5 Results and Discussion

5.1 Blood Glucose Variability and Prediction

Instead of implementing the Monte Carlo Markov Chain (MCMC) model to measure glucose variability, I implemented a few computationally less expensive versions of the MCMC code, which are functionally similar. These include:

- Moving average
- Regression line
- Spline

These approaches are used to obtain the best fit line through our the blood glucose data we collected. Obtaining a good stochastic model that models glucose fluctuations well first is pivotal in eventually deriving a glucose prediction algorithm. The plotted results for the same set of data are as follows:

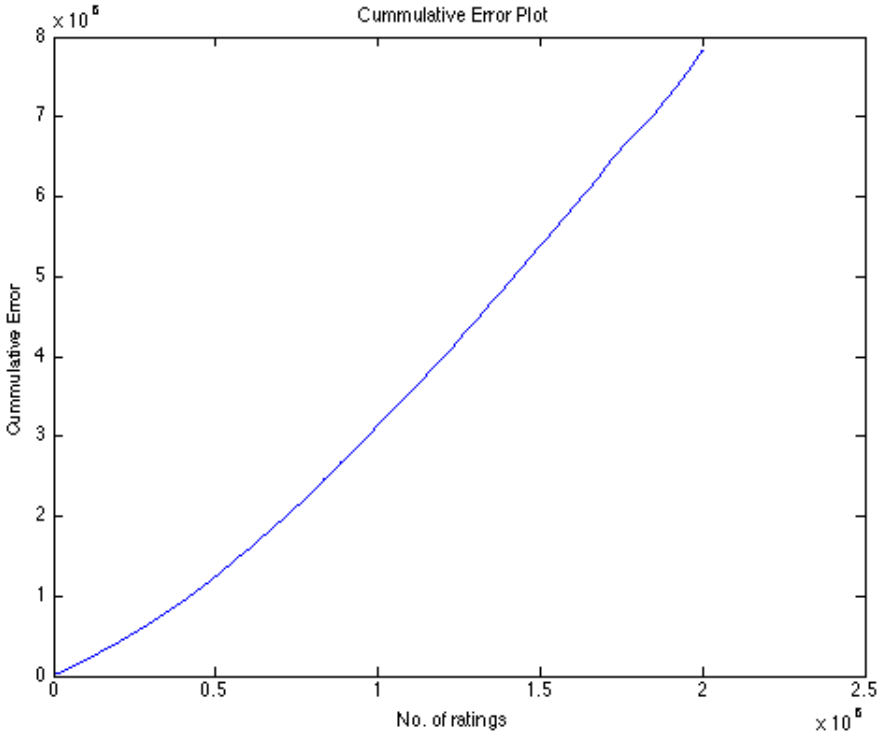


While the result plots from using moving average, local regression and spline follow the raw data well, it was inconclusive that an elegant, accurate stochastic model can be used to predict blood glucose fluctuations accurately. This is a challenging technical problem with government regulatory implications since it is questionable if our predictive model could pass government Food and Drug Administration (FDA) approvals.

5.2 Recommendation System Ratings

To measure the accuracy of the recommendation system, we plot the cumulative error as a function of the number of ratings. The measurement error is obtained by subtracting the estimated rating from the ground truth of our test set. A good recommendation system should have its cumulative

errors increase linearly as the number of ratings increases. For our movie database, the test set size is about 2 million ratings, while the training set size is about 8 million ratings. As shown from the graph, the plot is approximately linear as it scales with the number of ratings. Our results confirm previous work that Matrix factorization is a good approach for modeling recommendation systems.



In the broader context, our web application brings together a community of diabetics who share similar cultural and demographic backgrounds. This adds value to our customers because diabetes is a lonely and depressing disease. From our customer interview sessions, we learned from a diabetic, Kunal, that there are times he would worry about his high blood glucose levels because he did not know what was wrong in his lifestyle. He has no one to turn to, because only he knows himself best.

Our recommendation system is not just about reviewing restaurants, it can be used to recommend a diabetes-friendly lifestyle. For example, Diabeats can recommend diabetes-friendly food labels when you shop for groceries, or new glucose monitoring products in the market, based on a users' cultural and demographic background. We are bringing together a network of diabetics and recommend products and services relevant to them.

On another dimension, we do not have to limit our recommendation system to solve the diabetes problem. There are many other chronic diseases that can benefit from Diabeats. This include high blood pressure, which causes stroke) and high blood cholesterol, which causes heart disease.
By connecting diabetics together this way, we are building a relationship with our customers.We may not be able to compete against corporations who use their resources to invest in new technologies like non-invasive glucose monitoring. They may influence their market share by controlling product development. But what differentiates us, is we are controlling the market directly by being in an authoritative position to assess diabetic products and services.

5.3 Diabeats Facebook Marketing Campaign

A Facebook marketing campaign was conducted. Our advertisement is targeted at Facebook users who live in San Francisco, CA and whose age is 35 and older. The campaign started on April 17th 2012. Since then, our Facebook advertisement has received 31,129 impressions and 12 click throughs to our website, www.diabeats.co (Refer to Figure 3). The unexpected result was that out of these 12 click throughs, none of the prospects signed up with Diabeats. The hypothesis for this low conversion rate is that the user interface design for our website www.diabeats.co can be improved.

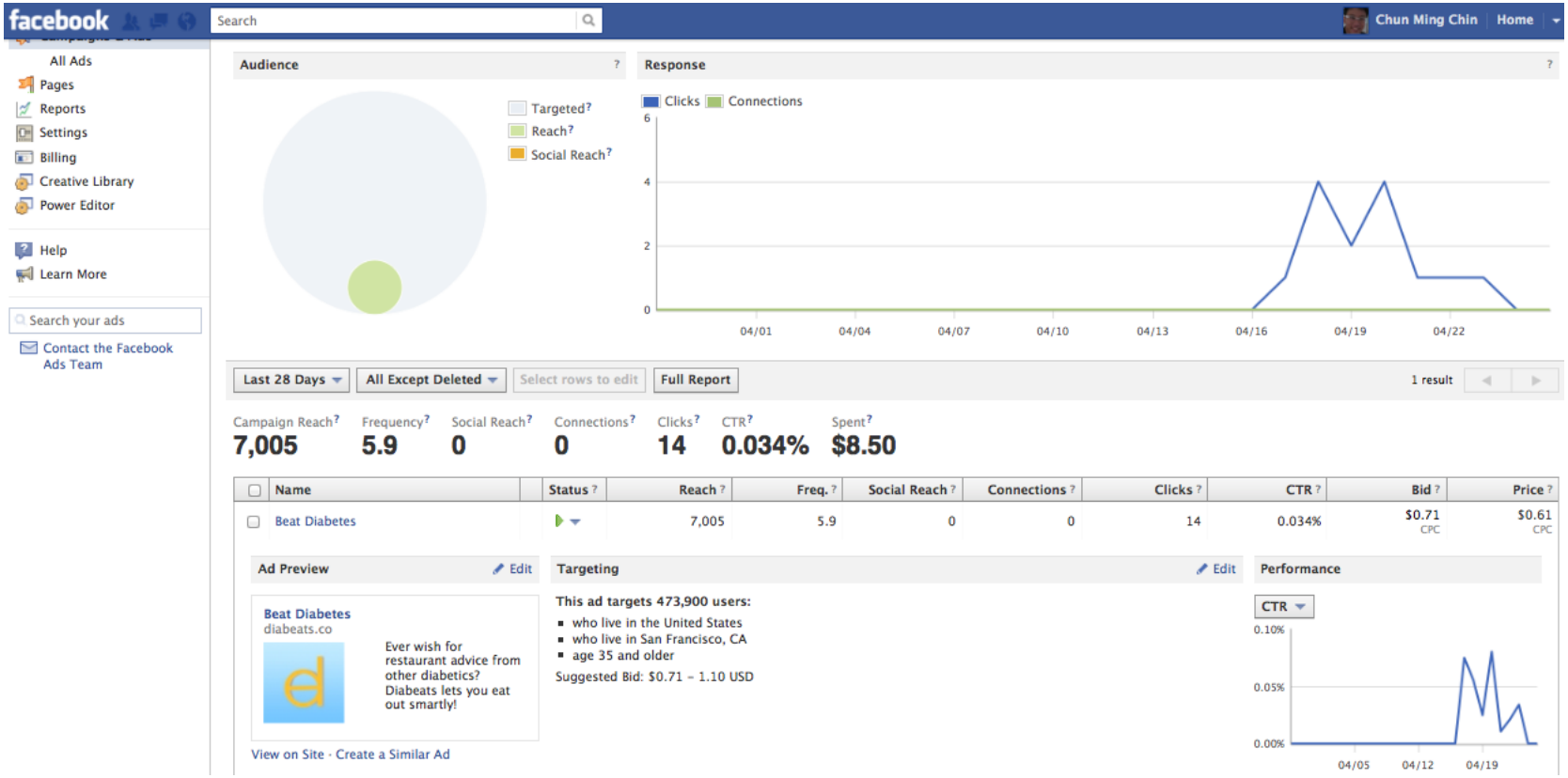


Figure 3: Facebook advertisement analytics

6 Conclusion/ Impact Statement

6.1 What was learned

6.1.1 Business-to-Business (B2B) Healthcare IT business models

The main opportunity in healthcare IT is in selling products and services to businesses entities (Hospitals, clinics etc.) rather than to individual consumers. In other words, healthcare ITs business model is based on business to business (B2B) instead of business to consumer (B2C). B2B is the predominant business model in healthcare IT because doctors are in an influential position to affect how well a product is adopted in the market. In order to buy in doctors, you should think about how your product/service reduces the legal liabilities of a doctor and reduces administrative cost. Improving the quality of healthcare is a secondary consideration [7].

6.1.2 Use Machine Learning To Solve Problems In Healthcare IT

There are many problems in healthcare that can be solved using machine learning. This is the study of how to uncover nontrivial dependencies between some observations, x, and a desired response y, for which a simple set of deterministic rules is not known. For example, in our first product iteration, we use observations such as carbohydrate intake and exercise intensity, to predict desired response - the glucose reading per unit time.

6.1.3 Large datasets are assets for any Company

Having access to relevant big data is key to building accurate and robust algorithms. This is the biggest disadvantage for start ups when they compete with large firms. A business model based on hardware is not feasible because big corporations have a lot more resources to invest in the development of hardware. In the diabetes monitoring industry, the trend is moving towards non-invasive continuous glucose monitoring devices (Refer to citation)

6.2 What remains to be learned

What remains to be learned is how early adopters would react to our Diabeats recommendation system. In addition, would our Matrix factorization implementation be powerful enough to provide

accurate recommendations? Otherwise, we may have to try other algorithm alternatives such as. Singular Value Decomposition or Convex Reformulation. Currently, we do not have any publicly available dataset to test our algorithm.

6.3 Weaknesses and Shortcomings

We lack the healthcare infrastructure to test our prediction algorithm, as well as a large patient medical record (i.e. glucose readings) database to train and improve the reliability of our prediction algorithm. In addition, we do not have easy access to clinicians who can conduct clinical trials - a necessary step to obtaining governmental approval for our healthcare application. These healthcare infrastructure are only available to corporations such as Siemens, Kaiser Permanante and Epic.

6.4 Strengths

We have developed proprietary technology (i.e. Prediction algorithm, recommendation system) that are potential intellectual property assets (i.e. patents). This asset would function as a barrier to entry, discouraging other competitors to enter the same field. This strength compensates for our weaknesses in having lack of access to patient medical records and a healthcare infrastructure to test our product, because out IP allows us to become attractive targets for acquisition by other companies.

6.5 Possible Applications

Our prediction algorithm can be used to track other chronic diseases besides Diabetes. Examples include Anemia, a blood disease characterized by low red blood cell count and high blood pressure. The condition of such diseases often depend on the lifestyle (Diet, exercise etc.) of the patient. Our restaurant recommendation system can be modified to recommend related products used by diabetics. These include recommending diabetic-friendly food options when shopping at the grocery store, or reviewing the performance of new diabetes monitoring hardware.

7 Acknowledgements

We would like to thank the following people for their help: The following conclusions on trends in the healthcare IT industry is based on my interviews with the following people and organizations:

- Donald Jones, Vice President of Qualcomm Wireless Health at San Diego, California, for his insights on trends in the healthcare IT industry.
- Paul Nerger, Chief Technical Officer of a Silicon Valley mobile healthcare start up, Happtique (www.happtique.com), for his insights on the business strategies used by mobile healthcare start ups in the United States.
- Alexander Smola, for his insights into machine learning which is pivotal for building the prediction algorithm and recommendation system.

8 References

Notes

[1]Prevention. D.O. (2007). National Diabetes Fact Sheet. Atlanta: CDC-INFO Contact Center: Centers for Diabetes Control and Prevention: http://cdc.gov/obesity/childhood/lowincome.html; Pesic, M. (2008, February 11).

[2]The Cost of Diabetes Medical Care. Retrieved June 28. 2010, from http://ezinearticles.com

[3]P. Magni, R. Bellazi. A Stochastic model to asses the Variability of Blood Glucose Time Series Diabetic Patients Self-Monitoring. IEEE Transactions on biomedical engineering, vol. 53, no. 6, June 2006.

[4]G.Sparacino, F. Zanderigo, S. Corazza, A. Maran, A. Facchinetti, C. Cobelli. Glucose Concentraion can be Predicted Ahead in Time From Continuous Glucose Monitoring Sensor Time Series. IEEE Transactions on biomedical engineering, vol. 54 no. 5. May 2007.

[5]Y.Koren. Collaborative Filtering with Temporal Dynamics. Yahoo Research.

[6]Scalable Machine Learning, Recommender Systems. Alex Smola. Yahoo! Research and ANU.

[7]Interview with Donald Jones, Vice President of Qualcomm Wireless Health

9 Appendices

9.1 Joint Stochastic Gradient Descent Matlab Code

```
% MovieID::MovieName::Genre
fidMovies = fopen('ml-10M100K/movies.dat');
dataMovies = textscan(fidMovies, '%d^%[^^]^%s');
save('dataMovies.mat', 'dataMovies');

% UserID::MovieID::Tag::Timestamp
fidTags = fopen('ml-10M100K/tags.dat');
dataTags = textscan(fidTags, '%d^%d^%[^^]^%d'); % Delimiter :: replaced by ^
save('dataTags.mat', 'dataTags');

% UserID::MovieID::Rating::Timestamp
fidRatings = fopen('ml-10M100K/ratings.dat');
dataRatings = textscan(fidRatings, '%d::%d::%d::%d');
save('dataRatings.mat', 'dataRatings');

load dataMovies.mat;
load dataRatings.mat;
load dataTags.mat;

matMovies = cell2mat(dataMovies(1)); % Convert cell to matrix
matRatings = cell2mat(dataRatings);
matRatingsSort = sortrows(matRatings,4);
bnd = floor(size(matRatingsSort,1)*0.8);

%% Extract trainning/ test sets
trainSet = matRatingsSort(1:bnd,:); % trainSet: First 80% ratings
testSet = matRatingsSort(bnd+1:end,:); % testSet: Last 20% ratings
```

```
%% Discard all movies/ratings in test set
testSetClean = testSet;
testSetClean(:,3) = 0;

%% Randomly permute instances in training set
y_um = trainSet(randperm(size(trainSet,1)),:); % Permute instances in dataset

numRatings = 8000043;
numUsers = max(y_um(1:numRatings,1));
numMovies = max(y_um(1:numRatings,2));

%% 4.2.1 Joint Stochastic gradient descent
% Keep user/movie ID labels since train data is permutated
% Train parameters v_u, v_m, b_u and b_m
% Loss function depends on v_u, v_m, b_u and b_m
timeSteps = 10;

v_ui = rand(numUsers, 50);
v_mj = rand(50, numMovies);
b_u = rand(numUsers, 1);
b_m = rand(numMovies,1);
v_uiLoss = zeros(numUsers, 50);
v_mjLoss = zeros(50, numMovies);
b_uLoss = zeros(numUsers);
b_mLoss = zeros(numMovies);

alpha = 1000;
beta = 1000;

lambda = 1; % Regularization factor is positive
h = waitbar(0,'Computing joint stochastic gradient descent');
```

```
for t = 1:timeSteps
    for r = 1:numRatings
        u = y_um(r,1); m = y_um(r,2);

        eta = 1/sqrt(alpha + (beta*t));
        p = b_u(u) + b_m(m) + v_ui(u,:)*v_mj(:,m) - y_um(r,3);

        % Estimate v_ui. u = 1...numUsers. i = 1... 50
        v_uiLoss(u,:) = double(p).*double(v_mj(:,m)') + lambda.*v_ui(u,:);
        v_ui(u,:) = v_ui(u,:) - eta*v_uiLoss(u,:);

        % Estimate v_mj. m = 1... numMovies. j = 1... 50
        v_mjLoss(:,m) = double(p).*double(v_ui(u,:)') + lambda.*v_mj(:,m);
        v_mj(:,m) = v_mj(:,m) - eta*v_mjLoss(:,m);

        % Estimate b_u. u = 1...numUsers
        b_uLoss(u) = double(p) + lambda.*b_u(u);
        b_u(u) = b_u(u) - eta*b_uLoss(u);

        % Estimate b_m. m = 1...numMovies
        b_mLoss(m) = double(p) + lambda.*b_m(m);
        b_m(m) = b_m(m) - eta*b_mLoss(m);

    end
    waitbar(t/timeSteps);
end
close(h);

%% 4.2.2 Report predicted ratings performance on test set
% Determine cumulative error with increasing no. of ratings
% Estimate rating yHat(u,m) for each user/movie pair
```

```
yHat = zeros(numUsers,numMovies);
for i=1:size(testSet,1)
    u = testSet(i,1); m = testSet(i,2);
    testSetClean(i,3) = b_u(u) + b_m(m) + v_ui(u,:)*v_mj(:,m); % (N x 50)*(50 x N)
end
errVect = (testSet(:,3) - testSetClean(:,3)).^2; %  Vectorized code
cumErr = cumsum(errVect);
plot(cumErr); % Plot cummulative error

%% 4.3 Movie attributes
% 4.3.2 Joint Stochastic gradient descent with new parameter v_c
% Addresses cold start problem
% Keep user/movie ID labels since train data is permutated
% Train parameters v_u, v_m, b_u and b_m
% Loss function depends on v_u, v_m, b_u and b_m

catAll =        {'Action',...
                 'Adventure',...
                 'Animation',...
                 'Children',...
                 'Comedy',...
                 'Crime',...
                 'Documentary',...
                 'Drama',...
                 'Fantasy',...
                 'Film-Noir',...
                 'Horror',...
                 'Musical',...
                 'Mystery',...
                 'Romance',...
                 'Sci-Fi',...
```

```
                    'Thriller',...
                    'War',...
                    'Western',...
                    }; % All categories

v_c = zeros(18,1);
TF = zeros(18,1);
count = zeros(18,1);

timeSteps = 10;

v_ui = .0001*rand(numUsers, 50);
v_mj = .0001*rand(50, numMovies);
v_c = .0001*rand(50,18);

b_u = .001*rand(numUsers, 1);
b_m = .001*rand(numMovies,1);
v_uiLoss = zeros(numUsers, 50); % v_u Loss
v_mjLoss = zeros(50, numMovies); % v_m Loss
v_cLoss = zeros(50, 18); % Sum v_c Loss
b_uLoss = zeros(numUsers);
b_mLoss = zeros(numMovies);

v_cSum = zeros(50,1);

alpha = 10000;
beta = 10000;

lambda = 0.001; % Regularization factor is positive
h = waitbar(0,'Computing joint stochastic gradient descent');
for t = 1:timeSteps
```

```
for r = 1:numRatings
    u = y_um(r,1); m = y_um(r,2);
    eta = 1/sqrt(alpha + (beta*t));

    % Compute v_c
    tmpStr = sprintf('%s', dataMovies{3}{r}); % Get string
    catMovie = dataread('string', tmpStr, '%s', 'delimiter', '|'); % Movie category
    for i=1:18 % Compare for each category
        for j = 1:size(catMovie, 1)
            TF(i) = strcmp(catAll{i}, catMovie{j}); % TF is scalar
            if (TF(i) == 1) % Don't waste time, compare next catMovie/catAll pair
                break;
            end
        end
    end

    for i = 1:18
        if (TF(i) == 1)
            v_cSum = v_cSum + v_c(:,i); % (50x1)
        end
    end

    p = b_u(u) + b_m(m) + v_ui(u,:)*v_mj(:,m) + v_ui(u,:)*v_cSum - y_um(r,3);

    for i = 1:18 % Update elements in specific category
        if (TF(i) == 1)
            v_cLoss(:,i) = double(p).*double(v_ui(u,:)') + lambda.*v_cSum; % 50x1
            if (r == 1)
                sum(v_cLoss(:,i));
            end
            v_c(:,i) = v_c(:,i) - eta*v_cLoss(:,i);
```

```
            end
        end


        % Estimate v_ui. u = 1...numUsers. i = 1... 50
        v_uiLoss(u,:) = double(p).*double(v_mj(:,m)' + v_cSum') + lambda.*v_ui(u,:);
        if (r == 1)
            sum(v_uiLoss(:,i))
        end
        v_ui(u,:) = v_ui(u,:) - eta*v_uiLoss(u,:);

        % Estimate v_mj. m = 1... numMovies. j = 1... 50
        v_mjLoss(:,m) = double(p).*double(v_ui(u,:)') + lambda.*v_mj(:,m);
        v_mj(:,m) = v_mj(:,m) - eta*v_mjLoss(:,m);

        % Estimate b_u. u = 1...numUsers
        b_uLoss(u) = double(p) + lambda.*b_u(u);
        b_u(u) = b_u(u) - eta*b_uLoss(u);

        % Estimate b_m. m = 1...numMovies
        b_mLoss(m) = double(p) + lambda.*b_m(m);
        b_m(m) = b_m(m) - eta*b_mLoss(m);

    end
    waitbar(t/timeSteps);
end
close(h);


fclose(fidMovies);
fclose(fidRatings);
```

```
fclose(fidTags);
```