A Bayesian Strategy to the 20 Question Game with Applications to Recommender Systems

by

Sunith Raj Suresh

Department of Statistical Science
Duke University

Date: _____

Approved:

_____
David L. Banks, Supervisor

_____
Merlise A. Clyde

_____
Cynthia Rudin

Thesis submitted in partial fulfillment of the requirements for the degree of
Master of Science in the Department of Statistical Science
in the Graduate School of Duke University
2017

Abstract

A Bayesian Strategy to the 20 Question Game with Applications to Recommender Systems

by

Sunith Raj Suresh

Department of Statistical Science
Duke University

Date: _____

Approved:

_____
David L. Banks, Supervisor

_____
Merlise A. Clyde

_____
Cynthia Rudin

An abstract of a thesis submitted in partial fulfillment of the requirements for the degree of Masters of Science in the Department of Statistical Science in the Graduate School of Duke University
2017

Abstract

In this paper, we develop an algorithm that utilizes a Bayesian strategy to determine a sequence of questions to play the 20 Question game. The algorithm is motivated with an application to active recommender systems. We first develop an algorithm that constructs a sequence of questions where each question inquires only about a single binary feature. We test the performance of the algorithm utilizing simulation studies, and find that it performs relatively well under an informed prior. We modify the algorithm to construct a sequence of questions where each question inquires about 2 binary features with AND conjunction. We test the performance of the modified algorithm via simulation studies, and find that it does not significantly improve performance.

Contents

List of Tables

List of Abbreviations and Symbols

Symbols

Z	A set of items
z_i	An element of Z
N_Z	Size of set Z
$\mathbf{z}^*$	User's latent choice
Q	A set of questions
N_Q	Size of set Q
d	Number of features for each item
$\mathbb{Z}_2$	Galois Field $\{0, 1\}$
z_{MAP}	represents item with Maximum a Posteriori
$z_{\text{top-3}}$	represents set of top 3 items ranked by decreasing order of Posterior Probability
$z_{\text{top-5}}$	represents set of top 5 items ranked by decreasing order of Posterior Probability

Acknowledgements

I would like to thank Professor David L. Banks, whose guidance and encouragement has been invaluable towards the development of my thesis.

1

Introduction

The 20 question game is played by two players, a questioner and an answerer. The answerer thinks of an item but does not reveal it. The goal of the questioner is to correctly guess the item by asking up to 20 questions about the item to which the answerer responds with correct answers. The original version of game involve only binary questions with 'true' or 'false' answers. Variations of the game involve multi-state answers including 'maybe', or allow the answerer to sometimes produce wrong answers (Rényi-Ulam game). In this paper, we consider the original version of the game and propose an algorithm that utilizes a Bayesian strategy to construct a sequence of questions in order to accurately determine the item.

We motivate this problem with an application to recommender systems. Consider a setup where a seller uses a website platform to market a set of items Z, where $|Z| = N_Z$. A user browses the website and maybe interested in a particular item or set of items denoted by $\mathbf{z}^*$. Note that the user's choice $\mathbf{z}^*$, is unknown to the seller, and hence is a latent choice. We utilize the strategy devised for the 20 question game to determine and pose a sequence of questions about the items to the user and utilize the answers provided by the user to best identify $\mathbf{z}^*$. Let Q denote the set of

questions that we consider, where $|Q| = N_Q$. Note one can reasonably assume that asking a user 20 questions on a website would negatively impact user experience. Accordingly we put a constraint on the number of questions we can pose to 5. There are a few interesting research inquires within this problem.

1. Given a set of question Q, there are $\frac{N_Q!}{(N_Q-5)!}$ possible sets of five questions we can ask. Among all these sets which one is optimal? What measure of optimality is appropriate?

2. Can we devise a non-greedy algorithm in order to determine the optimal set of questions efficiently? What is the computational complexity of such an algorithm?

3. Can we come up with a greedy algorithm that may not produce the optimal set of questions, but produces a set of questions that performs reasonably well? What is the accuracy of such an algorithm?

4. How well does the greedy algorithm perform in comparison to the non-greedy algorithm?

In this paper, we address the third question, and devise an algorithm that utilizes a Bayesian strategy to determine a sequence of questions. The algorithm is greedy and not guaranteed to determine the optimal sequence of question. We test the performance of the algorithm via a simulation study on a synthetic dataset.

2

Developing an Algorithm Utilizing a Bayesian Strategy

2.1 Binary Search and its Limitations

From results in information theory, we know that the with 20 bits, we can encode 2^{20} items (Shannon (1948)). So for, $N_Z = 2^{20} \approx 1.048^6$, we can determine the correct object in exactly 20 questions by employing a binary search algorithm that divides the item space in half, and at each node asks if the user's latent choice falls in one half of the item space. While such an algorithm is guaranteed to determine the correct answer in 20 questions, it is infeasible in recommender systems. Firstly, we constrain the number of questions we can ask to 5. Accordingly, such a binary search algorithm can determine the exact item only if $N_Z = 2^5 = 32$ which is unreasonably small for any modern e-commerce website. Secondly, even if we choose to expand the number of question we can ask, the first question in the binary search algorithm would ask the user to answer if their latent choice lies in a list of size of $\frac{N_Z}{2}$. For large N_Z, such a question would place a very high cognitive burden on the user to answer accurately. These limitations make a binary search approach infeasible for

this problem.

2.2 Algorithm Specification & Assumptions

Motivated to overcome the limitations of the binary search algorithm, we aim to develop an algorithm that can produce questions which are cognitively simple enough for the user to answer accurately, while at the same time able to divide the item space in half. Furthermore, for large N_Z, it is unlikely that we can determine the user's latent choice exactly using the information gathered from just 5 questions. So we seek to develop an algorithm that would allow us to quantify the uncertainty of our estimates by specifying a probability on them.

Note that items on a website can be described by its associated features. For instance if the items were books, then possible features would be genre, publishing date, fiction/non-fiction, won awards etc. Such features could be binary, categorical or continuous. Asking questions about the features of an item would be cognitively simple enough for a user to answer. So, now we have to determine which questions regarding features of an item we should ask in order to divide the item space in half. Before we describe the algorithm, we state some assumptions we make regarding the problem setup.

- Assume that each item can be uniquely represented by a set of d binary features. Accordingly $z_i \in \mathbb{Z}_2^d$. The items and features can be now represented by a binary matrix with N_Z rows and d features.

- Assume that there exist one item $\mathbf{z}^* \in Z$ that represents the user's latent choice. So, $|\mathbf{z}^*| = 1$

- Assume the questions we consider are binary. So for $q_j \in Q$, $q_j(Z) \in \{0, 1\}$ where 0 represents 'false' and 1 represents 'true'.

- Assume that the questions inquire only about a single feature of the item. So a typical question will be structured as "Does the desired item have feature $f_1 = 1$. True or False?"

- Assume that we can ask the user only up to 5 questions, and each question is answered correctly by the user.

2.3 An Algorithm Utilizing a Bayesian Strategy with Single Feature Questions

We develop an algorithm that utilizes a Bayesian strategy to determine a sequence of questions. The computational complexity of the algorithm is $\mathcal{O}(N_Z + d)$. The algorithm proceeds as follows.

1. There is a binary matrix Z whose rows represents items and columns represents features. The dimension of the matrix is $N_Z \times d$.

2. Construct a question set Q, where each question determines the state of a single feature. Accordingly $|Q| = N_Q = d$ and $q_j \in Q$ denotes a question regarding feature j.

3. Determine a prior probability $P(Z = z_k)$ for each item on the website, based on the user's demographic data and past buying behavior. For instance, a 20 year old male with past history of buying science fiction novels is more likely to prefer a science fiction book as opposed to a romantic comedy. For such a user, we would place higher prior probability on items whose features indicate science fiction and low prior probability on items whose features indicate romantic comedy.

4. For each question in the set Q , we can determine the probability thats its answer is true or false conditional on the prior probability as described below.

Note $q_j(z_k)$ represents the j^{th} feature of the k^{th} item

$$\begin{aligned} P(q_j(Z) = 1|P(z_k)) &= \sum_{k=1}^{N_Z} P(Z = z_k)\mathbb{1}[q_j(z_k) = 1], \\ P(q_j(Z) = 0|P(z_k)) &= 1 - P(q_j(Z) = 1|P(z_k)). \end{aligned} \tag{2.1}$$

5. Choose question q_j such that $P(q_j = 1|P(z_i))$ is closest to 0.5, and request the user to provide a binary response, 0 (False) and 1 (True). Since $P(q_j = 1|P(z_i))$ is close to 0.5 , the answer provided by the user will enable us to split the item space by nearly half.

6. On learning the answer, we can obtain posterior probabilities by simple reallocation.

$$\begin{aligned} P(Z = z_k|q_j(Z) = 1) &= \frac{P(Z = z_k)\mathbb{1}[q_j(z_k) = 1]}{\sum_{k=1}^{N} P(Z = z_k)\mathbb{1}[q_j(z_k) = 1]}, \\ P(Z = z_k|q_j(Z) \neq 1) &= \frac{P(Z = z_k)\mathbb{1}[q_j(z_k) \neq 1]}{\sum_{k=1}^{N} P(Z = z_k)\mathbb{1}[q_j(z_k) \neq 1]}. \end{aligned} \tag{2.2}$$

   Note that on asking a question and learning an answer from the user, several items will get a posterior probability of 0. The resulting mass will be reallocated proportionally over the other items.

7. We can then recompute the probability for each question in set Q conditional on the posterior probability, and repeat the steps 4-6 for the desired number of questions. The algorithm shall output a posterior probability for each item.

2.4 Recommendation Policy

The algorithm described above will output a posterior probability for each item in Z, of which many items will have a probability of 0. The item with the maximum a

posteriori (MAP) probability will be our best guess of z^*. In the case that there is a set of several items with equal MAP probability, our best guess would be uniform draw from this set. We believe it is more prudent to recommend a set of items, instead of just a single item. Such a recommendation policy would entail ranking the items by posterior probability and then recommending the top 3-5 items. Such a policy ensure that the user will view our best guess of z^* as well as other similar items, which would improve user experience and engagement.

3

Testing the Algorithm by Simulation Studies

In order to test our algorithm, we developed a simulation study which is described below.

3.1 Dataset

We generated a data matrix with N_Z items, each of which have d features. Typically the features have a correlation structure, so we first simulated normally distributed matrix with a correlation structure as follows

$$\begin{pmatrix} \vec{z_1} \\ \vdots \\ \vec{z_{N_Z}} \end{pmatrix} \sim N \left[\begin{pmatrix} 0 \\ \vdots \\ 0 \end{pmatrix}, \begin{pmatrix} 1 & \rho & \rho^2 & \rho^3 & \rho^4 & 0 & \cdots & 0 \\ \rho & 1 & \rho & \rho^2 & \rho^3 & \rho^4 & \cdots & 0 \\ \rho^2 & \rho & 1 & \rho & \rho^2 & \rho^3 & \cdots & 0 \\ \vdots & & & \ddots & & & & \vdots \\ 0 & \cdots & 0 & \rho^4 & \rho^3 & \rho^2 & \rho & 1 \end{pmatrix} \right]. \tag{3.1}$$

We set $\rho = 0.5$. In order to convert the matrix into binary form, a random value was chosen from each feature and values below it were set to 0 and values above it were set to 1.

3.2 Constructing an Informed Prior

Once the binary dataset was constructed, a random item from the data matrix was selected to be the user's latent choice, z^*. In order to construct the prior, we first placed a uniform probability on all items. We then assumed that we can accurately determine 5 features of z^* from user demographic information and past buying behavior. Those items which shared similar features with the 5 known features of z^* was then reallocated higher probability from those items with dissimilar features. Such a prior construction scheme will ensure that z^* will have a high prior probability among other items.

3.3 Simulation

We designed a simulation study on the generated dataset to evaluate the performance of the algorithm. Each iteration of the simulation performs the following steps.

1. Generate a binary matrix by randomly choosing a value from each feature setting values below it to 0 and values above it to 1.
2. Choose a random row from the matrix and denote its as z^*.
3. Construct a prior as described above.
4. Run the algorithm with 5 questions and determine the item with maximum a posteriori (z_{MAP}) as well as top-3 ($z_{\text{top-3}}$) and top-5 ($z_{\text{top-5}}$) items ranked by posterior probability.
5. Check if $\mathbf{z}^* = z_{\text{map}}$, if $\mathbf{z}^* \in z_{\text{top-3}}$ and if $\mathbf{z}^* \in z_{\text{top-5}}$.

4

Results From Simulation of Bayesian Algorithm with Single Feature Questions

We generated three different data matrices with the number of features, $d = 100$ and varying number of items, $N_Z = 75$ ($N_Z < d$), $N_Z = 100$ ($N_Z = d$) and $N_Z = 300$ ($N_Z > d$). We ran 1000 simulations on each of the data matrices. We first ran the simulations where the algorithm utilized an informed prior (described in Section 3.2). The results are described in Table 4.1.

Table 4.1: Results from Simulation of Bayesian Algorithm with Single Feature Questions using Informed Prior: Percent of Simulations for which $\mathbf{z}^* \in z_{\text{MAP}}$, $z_{\text{top-3}}$, $z_{\text{top-5}}$

Number of items (N_Z)	Number of features (d)	z_{MAP}	$z_{\text{top-3}}$	$z_{\text{top-5}}$
75	100	0.923	1	1
100	100	0.880	1	1
300	100	0.588	0.921	0.982

We then ran the simulations where the algorithm utilized an uniform prior. The results are described in Table 4.2.

We observe that for a fixed number of features (d), as the number of items

Table 4.2: Results from Simulation of of Bayesian Algorithm with Single Feature Questions using Uniform Prior: Percent of Simulations for which $\mathbf{z}^* \in z_{\text{MAP}}$, $z_{\text{top-3}}$, $z_{\text{top-5}}$

Number of items (N_Z)	Number of features (d)	z_{MAP}	$z_{\text{top-3}}$	$z_{\text{top-5}}$
75	100	0.440	1	1
100	100	0.341	0.961	1
300	100	0.112	0.348	0.572

(N_Z) grows, the performance of the algorithm generally degrades. When the algorithm utilizes an informed prior such that the latent choice $\mathbf{z}^*$ is assigned high prior probability, the performance of the algorithm is significantly better than when the algorithm utilizes a uniform prior. In our simulations we observe that even when $N_Z > d$, on utilizing an informed prior, the algorithm produces z_{MAP} exactly equal to $\mathbf{z}^*$ about 60% of the time. Further, the algorithm produces $z_{\text{top-5}}$ which includes $\mathbf{z}^*$ about 98% of the time.

5

Modifying the Algorithm to Pose Multiple Features Questions

One way to improve the algorithm is to relax the assumption that questions inquire only about a single feature of the items. Instead we can inquire about multiple features in a single question by combining them with boolean conjunctions AND, OR, NOT. Such questions would enable us to acquire information about multiple features from a single question which would allows us to gain more information from the user in 5 questions. Note that combining multiple features in a single question would make it more cognitively burdensome for the user to answer accurately. Accordingly we consider questions that combine at most two features. Since we still assume that the question are binary, we decided to consider only the AND conjunction.

Only a couple tweaks to the original algorithm is required to accommodate this change

- Step 1 should be modified to construct question set Q, where each question inquires about the state of two features via AND conjunction. So a typical question will be structured as "Does the desired item have features $f_1 = 1$

AND $f_2 = 1$. True or False?". Accordingly $|Q| = 4\binom{d}{2}$ and $q_{ij} \in Q$ denotes a question regarding feature i and feature j.

- Step 3 should be modified to compute the question probabilities as follows. Note $q_j(z_k)$ denotes the state of the j^{th} feature of the k^{th} item

$$
\begin{aligned}
P(q_{ij}(Z) = \{1,1\}|P(z_k)) &= \sum_{k=1}^{N_Z} P(Z = z_k)\mathbb{1}[q_i(z_k) = 1 \wedge q_j(z_k) = 1], \\
P(q_{ij}(Z) = \{1,0\}|P(z_k)) &= \sum_{k=1}^{N_Z} P(Z = z_k)\mathbb{1}[q_i(z_k) = 1 \wedge q_j(z_k) = 0], \\
P(q_{ij}(Z) = \{0,1\}|P(z_k)) &= \sum_{k=1}^{N_Z} P(Z = z_k)\mathbb{1}[q_i(z_k) = 0 \wedge q_j(z_k) = 1], \\
P(q_{ij}(Z) = \{0,0\}|P(z_k)) &= \sum_{k=1}^{N_Z} P(Z = z_k)\mathbb{1}[q_i(z_k) = 0 \wedge q_j(z_k) = 0].
\end{aligned}
\tag{5.1}
$$

The rest of the algorithm remains the same. The computational complexity of the modified algorithm is $\mathcal{O}(N_Z + d^2)$

6

Results From Simulation of Bayesian Algorithm with 2 Feature Questions

We ran 1000 simulations for the modified Bayesian Algorithm with 2 feature questions on the same three data matrices described in Chapter 4 . We first ran the simulations where the algorithm utilized an informed prior. The results are described in Table 6.1.

Table 6.1: Results from Simulation of Bayesian Algorithm with 2 Feature using Informed Prior: Percent of Simulations for which $z^* \in z_{\text{MAP}}$, $z_{\text{top-3}}$, $z_{\text{top-5}}$

Number of items (N_Z)	Number of features (d)	z_{MAP}	$z_{\text{top-3}}$	$z_{\text{top-5}}$
75	100	0.900	1	1
100	100	0.891	1	1
300	100	0.642	0.928	0.991

We then ran the simulations where the algorithm utilized an uniform prior. The results are described in Table 6.2.

Comparing the above simulation results (Table 6.1) to those obtained from the algorithm with single feature question (Table 4.1), we only observe a negligible im-

Table 6.2: Results from Simulation of Bayesian Algorithm with 2 Feature Questions using Uniform Prior: Percent of Simulations for which $z^* \in z_{\text{MAP}}$, $z_{\text{top-3}}$, $z_{\text{top-5}}$

Number of items (N_Z)	Number of features (d)	z_{MAP}	$z_{\text{top-3}}$	$z_{\text{top-5}}$
75	100	0.414	1	1
100	100	0.306	0.963	1
300	100	0.099	0.295	0.497

provement in performance from using the modified algorithm. Considering that the computational complexity of the original algorithm is linear and that of the modified algorithm is quadratic with respect to the number of features, the slight improvement in not significant.

7

Past and Future Work

7.1 Past Work

There are currently two popular 20 question games hosted online, which are operated by learning algorithms. An algorithm named 20q [1] developed by Robin Burgener, utilizes a neural network algorithm which has been patented (Burgener (2006)). The game allows the answer of think of any object that is not a proper noun, specific person, place or thing. Another website named Akinator [2] was developed by french company, Elokence. The website does allow users to think of a specific person or object, however the algorithm used by the website is currently held as a trade secret. Bayesian algorithms utilizing information theory have been developed for the 20 question game with noisy answers (Jedynak et al. (2012)). Within the field of computer science, much research has been done on "preference elicitation", which can be described as the problem of learning via queries. A survey of preference elicitation methods is provided in (Chen and Pu (2004)). Bayesian techniques for preference elicitation have been developed and described in (Guo and Sanner (2010)).

[1] http://www.20q.net/

[2] http://en.akinator.com/

7.2 Future Work

In this paper, we considered a simple dataset which was binary. One can incorporate categorical features by converting each category to a binary feature. Incorporating continuous features however is not straightforward, and warrants further work. It is unreasonable to assume, that the user knows the correct answer to binary questions regarding every feature of all item. Further work can be done on modifying the algorithm to allow the user to remain agnostic regarding a question, or sometimes answer a question incorrectly. The algorithm developed in this paper is step-wise greedy and not guaranteed to be optimal. Comparing the performance of this algorithm to a non-greedy algorithm is important to assess optimality and warrants further study.

Bibliography

Burgener, R. (2006), "Artificial neural network guessing method and game," US Patent App. 11/102,105.

Chen, L. and Pu, P. (2004), "Survey of Preference Elicitation Methods," .

Guo, S. and Sanner, S. (2010), "Multiattribute Bayesian Preference Elicitation with Pairwise Comparison Queries," in *Advances in Neural Networks - ISNN 2010, 7th International Symposium on Neural Networks, ISNN 2010, Shanghai, China, June 6-9, 2010, Proceedings, Part I*, pp. 396–403.

Jedynak, B., Frazier, P. I., and Sznitman, R. (2012), "Twenty questions with noise: Bayes optimal policies for entropy loss," *Journal of Applied Probability*, 49, 114–136.

Shannon, C. (1948), "A Mathematical Theory of Communication," *The Bell System Technical Journal*, 27, 379–423.