Automated Recommendation Systems
Collaborative Filtering Through Reinforcement Learning

Mostafa Afkhamizadeh
Department of MS&E,
Stanford University
Email: mafkhami@stanford.edu

Alexei Avakov
Department of Electrical Engineering,
Stanford University
Email: linrav@stanford.edu

Reza Takapoui
Department of Electrical Engineering,
Stanford University
Email: takapoui@stanford.edu

***Abstract*—Within this work we explore the topic of large scale, automated recommendation systems. We focus on collaborative filtering approaches, wherein a system suggests new products to users based on their viewing history as well as other known demographics. There are several approaches to this in current literature, the simplest of which treat it as a matrix completion problem. We explore the setting from a reinforcement learning perspective by applying traditional algorithms for reinforcement learning to the problem.**

I. Problem Formulation

Numerous online services such as Netflix, Amazon, Yelp, Pandora, online advertisings, etc. provide automated recommendations to help users to navigate through a large collection of items. Every time a user queries the system for a new item, a suggestion is made on the basis of the user's past history and (when available) their demographic profile. Two typical ways of producing these recommendations are collaborative filtering and content-based filtering. There are two simultaneous goals to be satisfied: helping the user to explore the available items and probing the user's preferences.

One of the models that captures this setting well is the multi-arm bandit, an important model for decision making under uncertainty. In this model a set of arms with unknown reward profiles is given and, at each time slot, the decision maker must choose an arm to maximize his expected reward. Clearly, the decision at each time slot should depend on previous observations. Thus, there is a trade-off between exploration, trying arms with more uncertain reward in order to gather more information, and exploitation, pulling arms with relatively high reward expectations.

For our purposes the arms have a very specific structure and this setting has previously been referred to as the "linear-bandits" model (see [3]). Here, it is assumed that the underlying matrix of preferences (which contains the rating user i gives to item j at entry (i, j)) has a low-rank structure. Hence, ratings made by user i to item j can be approximated by a scaler product of two feature vectors $a_i, b_j \in \mathbb{R}^p$, characterizing user and item respectively. In other words our observations, r_{ij} can be viewed as

$$r_{ij} = a_i^T b_j + z_{ij}$$

where z_{ij} represents the unexplained factors.

In the general setting, both the user and item feature vectors are treated as unknown, and our recommendation algorithms must estimate them over time. However, some works (like [1]) make a simplifying assumption that the item feature vectors are known. We explore both settings, but find more meaningful results in the case where the item feature vectors are known. In this case, the item feature vectors can be either constructed explicitly, or derived from users' feedback using matrix factorization methods. With the item latent vectors in hand, we can treat each user independently and throughout the explore-exploit trade-off, we can try to estimate and exploit the users latent vectors. These feature vectors can depend on users' demographic information and their past behavior in rating items.

The goal of our system is to develop a recommendation policy, which suggests items to users. This policy will, at each time slot, output a recommendation based on the previous observations. This policy must properly adjust for the explore-exploit trade-off, and classically there are two types of policies, which differ in the way they perform exploration: optimistic policies, e.g. upper confidence bound (UCB), and probabilistic policies, e.g. posterior sampling. UCB algorithms have been applied to this problem in the past, but posterior sampling is less common. Posterior sampling (also known as Thompson sampling) was introduced in 1933 and offers significant advantages over UCB methods (as shown in [2]), however until recently it has not been very popular or feasible.

Our primary objective is to explore the feasibility of collaborative filtering through posterior sampling. We analyze its performance on real world data, specifically the freely available MovieLens datasets, and compare it to existing methods such as UCB and the work done in [1]

II. System Models and Algorithms

In this section we will introduce some notation used throughout the rest of this work, as well as the algorithms that we seek to implement.

A. Notation

We have a set of users, $i = 1, 2, \ldots, m$, with corresponding feature vectors $a_i \in \mathbb{R}^p$; and items, $j = 1, 2, \ldots, n$, with corresponding feature vectors $b_j \in \mathbb{R}^p$. We refer to these feature vectors collectively as $A \in \mathbb{R}^{p\times m}$ and $B \in \mathbb{R}^{p\times n}$, thus the "true" ratings can be captured in the matrix $A^T B$. At each time $t \in \mathbb{Z}^+$, a user $i^{(t)}$ will enter the system and be recommended an item $j^{(t)}$, after which they will give it a rating $r^{(t)}$ according to

$$r^{(t)} = a_{i^{(t)}}^T b_{j^{(t)}} + z^{(t)}$$

Where $z^{(t)}$ captures the unexplainable deviation of the observation from our model. We refer to the viewing history at time t as the sequence $\mathcal{H}^{(t)} = \left\{\left(i^{(\tau)}, j^{(\tau)}, r^{(\tau)}\right)\right\}_{\tau=1}^{t-1}$, i.e. all the viewings in the system before time t. Thus on a high-level, at time t our program seeks to use its knowledge of user $a_{i^{(t)}}$ to make the best possible recommendation.

The job of a recommendation system is to define a function $\mu_H(\cdot)$, which given a user will output a recommendation for that user. Unknown to the system, there is some optimal policy which at each time t would output recommendation $j_*^{(t)}$. To measure the performance of our system, we will compare the system's recommendations to the best recommendation. Specifically define the regret of the system, at time t, to be

$$R(t) = \sum_{\tau=1}^{t} \left(a_{i^{(\tau)}}^T b_{j_*^{(\tau)}} - \mathbb{E}\left[r^{(\tau)}\right]\right)$$

That is, at each time-step we increase our regret by how far the expected rating of our recommendation differs from the best possible rating. Ultimately we seek to derive a policy which achieves minimal regret.

B. Posterior Sampling

Algorithm 1 Posterior Sampling

```
Start with prior distribution on $(A, B)$, $f(A, B)$
for $t = 1, 2, \ldots$ do
    observe arrival of user $i^{(t)}$
    sample $\hat{A}, \hat{B} \sim f(A, B|\mathcal{H}^{(t)})$
    compute and output recommendation $j^{(t)}$ where
        $j^{(t)} = \arg\max_j \mathbb{E}\left[\hat{a}_i^T \hat{b}_j\right]$
    .
    observe the user's rating $r^{(t)}$
end for
```

The idea behind posterior sampling algorithm is to force optimism through probabilistic action. Specifically at each time step, t, we will make a recommendation $j^{(t)}$ based on the probability that it is the best possible recommendation, $\mathbb{P}\left(j^{(t)} = j_*^{(t)}\right)$. However, this probability is inaccessible, so instead the algorithm samples a model for the unknown feature vectors based on the probability that they are the true feature vectors (given the viewing history), and finds the optimal recommendation should this be the true model. It can be shown that this sampling technique is equivalent to sampling a recommendation based on the probability it is optimal, and a more detailed description of the algorithm and its motivations can be seen in [2]. Thus the algorithm proceeds to keep track of the distribution of model parameters at each time step, and updates them accordingly.

To implement this algorithm all that remains is to choose a prior on the model parameters, and compute their posterior distribution given a viewing history. As in [3], and other prior literature, we assume $a_i, b_j \sim \mathcal{N}\left(0, \mathbb{I}_p/p\right)$ i.i.d.. Furthermore we assume that unexplained deviations of the observations are Gaussian, i.e. $z^{(t)} \sim \mathcal{N}(0, \sigma_z^2)$. Now we are ready to compute the posterior distribution.

Using Bayes' rule observe (for compactness we use $f(\cdot)$ to denote the distribution of the argument):

$$\begin{aligned} f\left(A, B \,\middle|\, \mathcal{H}^{(t)}\right) &= \frac{f\left(\mathcal{H}^{(t)}\middle| A, B\right) f(A) f(B)}{f\left(\mathcal{H}^{(t)}\right)} \\ &= \frac{f(A) f(B) \prod_{\tau=1}^{t-1} f\left(z^{(\tau)}\right)}{\int_{A,B} f(A) f(B) \prod_{\tau=1}^{t-1} f\left(z^{(\tau)}\right) \mathrm{d}A \mathrm{d}B} \end{aligned}$$

In the above $z^{(\tau)} = r^{(\tau)} - a_{i^{(\tau)}}^T b_{j^{(\tau)}}$, and the integral of the denominator is over the entire space of $\mathbb{R}^{p\times m} \times \mathbb{R}^{p\times n}$.

For the rest of this report, we consider the simpler case where the vectors b_j are given and we treat each user independently. This problem is extensively studied in literature, but as far as we can tell has never been solved or analyzed through posterior sampling. We explore it more concretely below.

For compactness we will consider only the feature vector of a single user, $a \in \mathbb{R}^p$, a priori we assume it comes from $\mathcal{N}(0, \mathbb{I}_p/p)$ as above, and we now consider the viewing history $H^{(t)}$ to be the history of the active user (as opposed to all users). We can now compute the posterior distribution as follows:

$$\begin{aligned} f_{a|\mathcal{H}}\left(a \,\middle|\, \mathcal{H}^{(t)}\right) &= \frac{f_{\mathcal{H}|a}\left(\mathcal{H}^{(t)}\middle| a\right) f_a(a)}{f_{\mathcal{H}}\left(\mathcal{H}^{(t)}\right)} \\ &= \frac{f_a(a) \prod_{\tau=1}^{t-1} f_z\left(z^{(\tau)}\right)}{\int_{\mathbb{R}^p} f_a(a) \prod_{\tau=1}^{t-1} f_z\left(z^{(\tau)}\right) \mathrm{d}a} \\ &= \frac{f_a(a) \prod_{\tau=1}^{t-1} f_z\left(r^{(\tau)} - a^T b_{j^{(\tau)}}\right)}{\int_{\mathbb{R}^p} f_a(a) \prod_{\tau=1}^{t-1} f_z\left(r^{(\tau)} - a^T b_{j^{(\tau)}}\right) \mathrm{d}a} \end{aligned}$$

But observe in this simple case computing the posterior is much simpler. The numerator is clearly a Gaussian, and the denominator is just a normalizing term, thus we determine $a \,\middle|\, \mathcal{H}^{(t)} \sim \mathcal{N}\left(\mu_a^{(t)}, \Sigma_a^{(t)}\right)$. We can formulate a recursive update rule for the parameters $\mu_a^{(t)}, \Sigma_a^{(t)}$ by massaging the numerator into an appropriate form (this is done in the appendix). We find the following update equations for the posterior:

$$\begin{aligned} \Sigma_a^{(t)} &= \left(\Sigma_a^{(t-1)} + \frac{b^{(t-1)} b^{(t-1)^T}}{\sigma_z^2}\right)^{-1} \\ \mu_a^{(t)} &= \Sigma_a^{(t)} \left(\Sigma_a^{(t-1)^{-1}} \mu_a^{(t-1)} + \frac{r^{(t-1)}}{\sigma_z^2} b^{(t-1)}\right) \end{aligned}$$

These recursive update equations are convenient for implementation, and can be used efficiently by storing Σ^{-1}, however some intuition as to their operation can be seen by applying the matrix inversion lemma. Through it, we find:

$$\begin{aligned}\Sigma_a^{(t)} &= \Sigma_a^{(t-1)} - \frac{\Sigma_a^{(t-1)} b_{j(t-1)} b_{j(t-1)}^T \Sigma_a^{(t-1)}}{\sigma_z^2 + b_{j(t-1)}^T \Sigma_a^{(t-1)} b_{j(t-1)}} \\ \mu_a^{(t)} &= \mu_a^{(t-1)} + \frac{r^{(t-1)}\Sigma_a^{(t-1)} - \Sigma_a^{(t-1)} b_{j(t-1)} \mu_a^{(t-1)^T}}{\sigma_z^2 + b_{j(t-1)}^T \Sigma_a^{(t-1)} b_{j(t-1)}} b_{j(t-1)}\end{aligned}$$

Thus, essentially, at each step the posterior mean shifts towards (or away) the feature vector of the recommended item. Similarly the covariance Σ thins out to select for single direction.

The rest of our work revolves mostly around analyzing the simplified problem setting, however this simplification is extremely useful for the general case as well. Observe,

$$f\left(A, B \,\middle|\, \mathcal{H}^{(t)}\right) = f\left(A \,\middle|\, B, \mathcal{H}^{(t)}\right) f\left(B \,\middle|\, \mathcal{H}^{(t)}\right)$$

Thus we can perform posterior sampling in the general case by first sampling item features $\hat{B}$, according to $f\left(B \,\middle|\, \mathcal{H}^{(t)}\right)$, and then sampling A from a gaussian distribution with mean and variance determined by the previously derived update equations given the selected features $\hat{B}$. Unfortunately the distribution of $B \,|\, \mathcal{H}^{(t)}$ is quite complicated; after vectorizing the matrix B into a vector $\overrightarrow{B} \in \mathbb{R}^{np}$ we find:

$$f\left(\overrightarrow{B} \,\middle|\, \mathcal{H}^{(t)}\right) \propto \left(p\left(\overrightarrow{B}\right)\right)^{-1/2} \exp\left(k \overrightarrow{B}^T \overrightarrow{B} + c^T \overrightarrow{B}\right)$$

In the above p is a polynomial function in the entries of $\overrightarrow{B}$, k is some scalar, and c is a vector in $\mathbb{R}^{np}$. Unfortunately, even in this form, it is still unclear how to sample from this distribution.

C. A UCB Approach

Algorithm 2 UCB

```
Start with prior distribution on $(A, B)$, $f(A, B)$, and an optimism parameter $p \in (0, 1)$
for $t = 1, 2, \ldots$ do
    Observe arrival of user $i^{(t)}$
    Compute the distribution on the reward of each item
    For all items, compute $U_j$, the $p$-th percentile of the reward of item $j$
    Compute and output recommendation $j^{(t)}$ where
        $j^{(t)} = \arg\max_j U_j$
    Observe the user's rating $r^{(t)}$
end for
```

UCB is a completely different approach from posterior sampling. At each timestep the algorithm computes an upper confidence bound on the reward of each of the items. The algorithm will then suggest the algorithm with the highest UCB. For our purposes, we will use a specific percentile of the reward as the UCB of each item. This is generally hard to do and other literature uses various heuristics to determine a UCB. In the general problem setting, it is unclear how to implement UCB in any meaningful way, however it is rather elegant in the simplified case of given item feature vectors.

In the simplified case, (using the priors described in the previous section) we observe that the posterior of a given $\mathcal{H}^{(t)}$ is Gaussian, thus the distribution on the reward of recommending item j is also Gaussian. We compute the mean and variance as follows:

$$\begin{aligned}\sigma_j^2 &= b_j^T \Sigma_a b_j + \sigma_z^2 \\ \mu_j &= b_j^T \mu_a\end{aligned}$$

Thus computing the p-th percentile of the reward can be done, simply by inverting the cdf of the normal distribution.

D. Mixed Approaches

From evaluation we observe that UCB and posterior sampling each have unique advantages. Thus we propose various schemes that allow you to achieve the various performance trade-offs of both. First we propose an ϵ-greedy approach and second we propose a two-phase approach. These were both studied in the simplified case, but could potentially be applied to the general setting as well.

The ϵ-UCB algorithm will flip a weighted coin at each timestep to decide weather to obtain a recommendation through posterior sampling or through UCB. Specifically the algorithm will elect to perform UCB ϵ percent of the time.

The two phased approach will begin by learning through posterior sampling until some time T, after which it proceeds to output recommendations through the UCB approach.

In the next section, we will thoroughly study the performance of all of the algorithms presented in this section.

E. The Case of No-repeat Recommendations

Throughout this work we assumed that it is relevant to recommend the same item several times. However, in some settings this is not very natural. For instance, if the system provides recommendations for viewing movies, all of the above algorithms would eventually chose to show the same movie over and over. Clearly this is not very useful, and this can be resolved in several ways. We could lower the reward of successive viewings, but this adds a complicated time dependence to our model. More simply we can prohibit the algorithm from suggesting the same item multiple times. In the case of suggesting movies this is natural since users would rarely view the same production multiple times.

III. Implementation and Evaluation

In this section, we present our implementation results for the aforementioned algorithms. For the purpose of numerical simulations, we used MATLAB. We have carried out algorithms both on synthetic data and freely available MovieLens dataset.

For the purpose of synthetic data, we generate a random 943×1682 matrix (the same size as MovieLens data) with rank 30 by generating random Gaussian feature matrices and multiplying them together. Each of the entries of feature matrices comes from a $\mathcal{N}(0, 1/p)$. Then, we will take item feature vectors as granted and try to estimate user feature matrix by considering a Gaussian prior.

Figure 1 shows the cumulative regret versus time, for posterior sampling and UCB with four different parameters. We can see that posterior sampling might work worse at first by exploring too much, but it pays off later when the better understanding of the arms comes to help later.

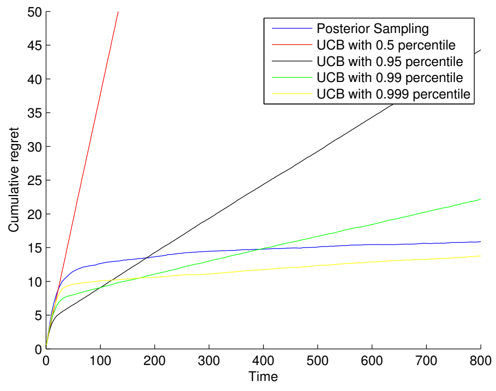

Fig. 1. Cumulative regret of posterior sampling and UCB algorithms on synthetic data

Notice that the regret observed by each of these algorithms is very good compared to the total reward they obtain (the cumulative rewards at t=800 is in the order of 500 while the difference in rewards is in the order of 1). In order to show this, we have plotted the cumulative reward versus time for all of these algorithms in Figure 2 and it can be seen that it is close to the optimal reward.

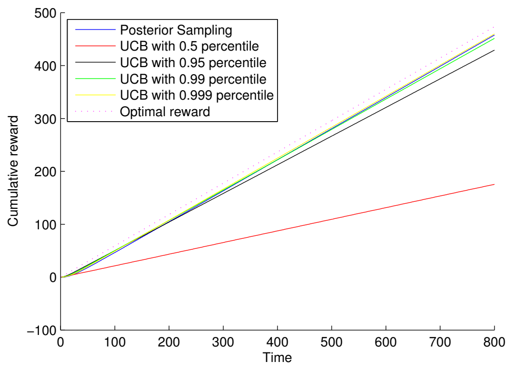

Fig. 2. Cumulative reward of posterior sampling and UCB algorithms on synthetic data

We also tried similar simulations for different ranks of the underlying matrix. Figure 3 shows the performance of these algorithms when the rank of the preference matrix is 100. It can be seen that posterior sampling outperforms UCB. One interesting observation is that unlike posterior sampling, UCB methods are very sensitive to the parameters used in algorithms (in our case the percentile parameter), and using inappropriate parameter may result in non-zero asymptotic regret. We observe that the optimal tuning is highly sensitive to the data, specifically to its rank.

As a variation of the introduced algorithms, we have carried out a combination of posterior sampling and greedy

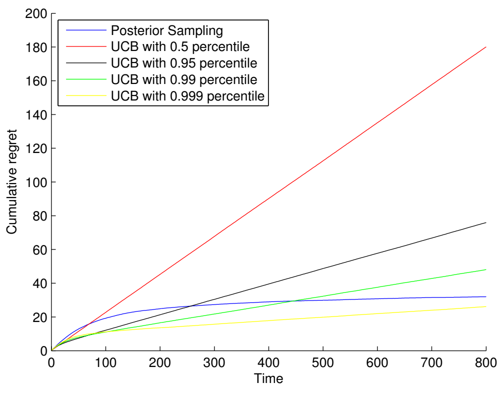

Fig. 3. Cumulative regret of posterior sampling and UCB algorithms for synthetic data with rank = 100

algorithms. Here, the greedy algorithm chooses the arm that maximized the expected instantaneous reward and can be considered as UCB with percentile 50%. At each time, the ϵ-greedy algorithm makes greedy decision with probability ϵ, and performs an iteration of posterior sampling with probability $1-\epsilon$. By looking at Figure 4, we can see that the performance of the greedy algorithm improves dramatically when it's combined with posterior sampling 50% of the time. This will result in even more computationally efficient methods while the regret still remains acceptably low.

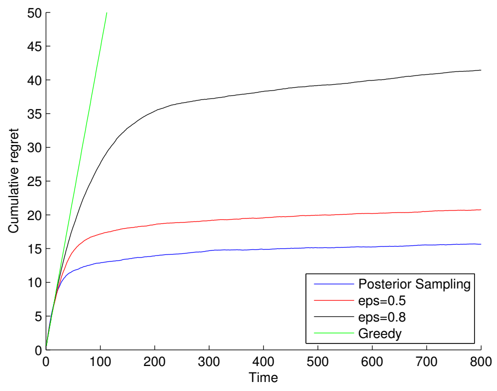

Fig. 4. Cumulative regret for hybrid approach on synthetic data

We also carried out the posterior sampling and UCB algorithms under the assumption that no item can be recommended to a user more than once. Notice that in this case, we expect the regret to be decreasing at some point, because the expected regret at time 1682 is equal to zero. Figure 5 shows the performance of these algorithms in this case.

We have implemented all these methods for the MovieLens dataset and got similar results. For example Figure 6 shows the cumulative regret for the posterior sampling and UCB with different parameters on MovieLens dataset. As seen in Figure 6, UCB algorithm with parameter 0.95 works better than the other instances of UCB, which further shows that there is no rule for finding the best parameters for UCB algorithms.

IV. Conclusions

All of the algorithms we analyzed perform extremely well. The difference in regret between them is negligible compared

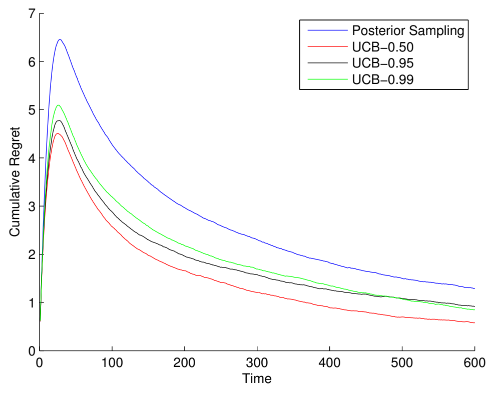

Fig. 5. Cumulative regret for the case of no repetitions, on synthetic data

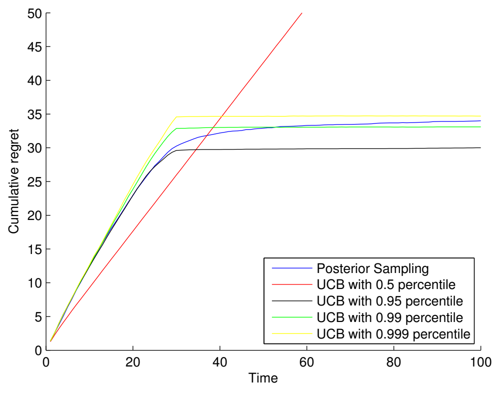

Fig. 6. Cumulative regret of posterior sampling and UCB on MovieLens dataset

to the total reward collected. Thus we advocate posterior sampling as the best general purpose solution for several reasons. First, it is extremely efficient compare to the UCB style approach. Second, it does not require any tuning; while we observed that UCB can outperform posterior sampling it is extremely reliant on proper tuning (which can be hard to determine in practice). Furthermore, posterior sampling can clearly be extended to the general problem setting, whereas our stated UCB method is not. Lastly we note that using previously mentioned hybrid approaches it is possible to achieve many different efficiency/regret trade-offs.

V. Future Work

It would be interesting to more closely analyze the general case. Posterior sampling can be implemented through Gibb's sampling, or the Metropolis-Hastings algorithm. UCB as described in this paper would be much more difficult to implement, but we could try various heuristics and other UCB style algorithms.

Alternatively this work could be continued in the practical direction by building a real-life recommendation system utilizing these algorithms and studying its performance.

Appendix

A. Derivation of Posterior Update Rules

Again consider the simplified case where we know the latent feature vectors of the items. For compactness we will consider only the feature vector of a single user, $a \in \mathbb{R}^p$. Then:

$$\begin{aligned} f\left(a \middle| \mathcal{H}^{(t)}\right) &= \frac{f_a(a)\prod_{\tau=1}^{t-1} f_z\left(r^{(\tau)} - a^T b_{j^{(\tau)}}\right)}{\int_{\mathbb{R}^p} f_a(a)\prod_{\tau=1}^{t-1} f_z\left(r^{(\tau)} - a^T b_{j^{(\tau)}}\right) \mathrm{d}a} \\ &= C_1 f_a(a) \prod_{\tau=1}^{t-1} f_z\left(r^{(\tau)} - a^T b_{j^{(\tau)}}\right) \\ &= C_2 f_z\left(r^{(t-1)} - a^T b_{j^{(t-1)}}\right) f\left(a \middle| \mathcal{H}^{(t-1)}\right) \\ &= C_3 \exp - \frac{\left(r^{(t-1)} - a^T b_{j^{(t-1)}}\right)^2}{2\sigma_z^2} \times \\ &\quad \exp -\frac{1}{2}(a - \mu_a^{(t-1)})^T {\Sigma_a^{(t-1)}}^{-1} (a - \mu_a^{(t-1)}) \end{aligned}$$

Now it is clear that the distribution remains Gaussian. At this point simply compute the coefficients of the quadratic and linear terms to solve for the new mean and covariance. This yields

$$\begin{aligned} \Sigma_a^{(t)} &= \left(\Sigma_a^{(t-1)} + \frac{b^{(t-1)} {b^{(t-1)}}^T}{\sigma_z^2}\right)^{-1} \\ \mu_a^{(t)} &= \Sigma_a^{(t)} \left({\Sigma_a^{(t-1)}}^{-1} \mu_a^{(t-1)} + \frac{r^{(t-1)}}{\sigma_z^2} b^{(t-1)}\right) \end{aligned}$$

B. Woodury Matrix Identity & Update rules

Recall the Woodury Matrix Identity:

$$(A + UCV)^{-1} = A^{-1} - A^{-1}U\left(C^{-1} + VA^{-1}U\right)^{-1} VA^{-1}$$

For ease of notation in this section we will refer to $\Sigma_a^{(t)}$ as Σ_t, $\mu_a^{(t)}$ as μ_t, $r^{(t)}$ as r, and lastly we refer to $b^{(t)}$ simply as b. Apply the lemma to the previously derived update rules:

$$\begin{aligned} \Sigma_t &= \left(\Sigma_{t-1} + \frac{bb^T}{\sigma_z^2}\right)^{-1} \\ &= \Sigma_{t-1} - \frac{\Sigma_{t-1} bb^T \Sigma_{t-1}}{\sigma_z^2 + b^T \Sigma_{t-1} b} \end{aligned}$$

Now we can plug this into the derivation of μ_t:

$$\begin{aligned} \mu_t &= \Sigma_t \left({\Sigma_{t-1}}^{-1} \mu_{t-1} + \frac{r}{\sigma_z^2} b\right) \\ &= \left(\Sigma_{t-1} - \frac{\Sigma_{t-1} bb^T \Sigma_{t-1}}{\sigma_z^2 + b^T \Sigma_{t-1} b}\right)\left({\Sigma_{t-1}}^{-1}\mu_{t-1} + \frac{r}{\sigma_z^2} b\right) \\ &= \mu_{t-1} - \frac{\Sigma_{t-1} b \mu_{t-1}^T b}{\sigma_z^2 + b^T \Sigma_{t-1} b} + \\ &\quad \frac{r}{\sigma_z^2}\left(\frac{\sigma_z^2 \Sigma_{t-1} b + b^T \Sigma_{t-1} b \Sigma_{t-1} b - \Sigma_{t-1} bb^T \Sigma_{t-1} b}{\sigma_z^2 + b^T \Sigma_{t-1} b}\right) \\ &= \mu_{t-1} + \left(\frac{r\Sigma_{t-1} - \Sigma_{t-1} b \mu_{t-1}^T}{\sigma_z^2 + b^T \Sigma_{t-1} b}\right) b \end{aligned}$$

References

[1] Yash Deshpande, Andrea Montanari, "Linear Bandits in High Dimension and Recommendation Systems". Available online: http://arxiv.org/abs/1301.1722

[2] Daniel Russo, Benjamin Van Roy, "Learning to Optimize Via Posterior Sampling". Available online: http://arxiv.org/abs/1301.2609

[3] Paat Rusmevichientong, John N. Tsitsiklis, "Linearly Parameterized Bandits". Available online: http://arxiv.org/abs/0812.3465