A Bayesian Value–Iteration Approach to Contextual Multi-Armed Bandits

October 27, 2025

Abstract

We present a Bayesian value-iteration framework for contextual multi-armed bandit problems that treats the agent's posterior distribution as the state of the Markov Decision Process. We apply finite dimensional priors on the unknown reward parameters, and the context transition kernel. Value Iteration on this MDP yields an optimal policy for the contextual multi-armed bandit problem.

We first revisit classical Bernoulli bandits, where our formulation gives the same solution as the Gittins index. We then extend the problem to a toy contextual problem, and finally demonstrate that our solution also applies in a airline seat pricing simulation. Across all experiments, the proposed Contextual Value Iteration (CVI) algorithm achieves higher rewards compared to our baseline algorithms, while remaining computationally manageable for small to medium problem sizes. We discuss limitations coming from the curse of dimensionality, and directions to overcome these through function approximation techniques. Furthermore, our approach only works if the right priors can be chosen. Without a good choice of priors, the approach breaks down.

1 Introduction

The multi-armed bandit (MAB) problem, first posed by Robbins 1952, captures the tension between learning and earning when decisions must be taken sequentially. In the classical, context-free formulation, an elegant solution is provided by the Gittins index (J. Gittins 1974; J. C. Gittins 1979; J. Gittins, Glazebrook, and Weber 2011): each arm receives an index that depends only on its own history, and pulling the arm with the largest index is optimal under geometric discounting.

Real applications, however, are rarely context-free. Consider airline seat pricing: the demand curve changes with the day of week, the remaining time to departure, and competitor prices. That context is shared across all prices, so a sale at one price tells something about every other price. Once such coupling is present, the separate-index trick breaks down.

We propose a simple fix: treat everything uncertain — including the context dynamics themselves—in a fully Bayesian manner. The agent's belief at time t is then just another state variable. Learning then becomes an ordinary MDP problem, solvable by value iteration. The approach is conceptually straightforward, can incorporate a wide range of approaches, and is shown to perform well in practice.

Contributions

1. We formalise a single Bayesian MDP whose belief state contains both the pay-off parameters and the context kernel. Without context, the model reproduces the Gittins index, but can be extended to incorporate context as well.
2. We also show that the model can be practically implemented as the Contextual Value Iteration (CVI) algorithm, and show in simulations that CVI outperforms contextual ε-greedy and Thompson sampling on small–to–medium sized problems.

2 Background and related work

2.1 Background

The MAB problem is a classic decision-making dilemma, introduced in Robbins 1952, in which an agent must choose between several competing options, each with unknown reward distributions, to maximize cumulative reward over time. Common methods for addressing this problem rely on heuristics to balance exploration and exploitation. For instance the epsilon-greedy approach, in which a proportion of choices a random choice is made (Cesa-Bianchi and Fischer 1998 provides more insight in the performance of this approach). Another standard approach is the Upper Confidence Bound (UCB) algorithm, in which an optimistic upper estimate of results is used (starting from Lai and Robbins 1985, then described more specifically in Auer, Cesa-Bianchi, and Fischer 2002). And Thompson Sampling Thompson 1933, in which a random sample from the posterior pay-off distribution is used as estimate for the pay-off. More advanced algorithms almost all use one of these approaches as a basis. Lai and Robbins 1985 shows that regret grows at least logarithmically, so an algorithm is said to solve the MAB problem if it can match this lower bound. In Kuleshov and Precup 2014 an extensive review is available showing both how standard algorithms work, and reviewing practical performance comparisons in various scenarios.

Kuleshov and Precup 2014 show through a range of experiments that when considering performance of an algorithm on the MAB problem, the relevant characteristics of the problem that influence the performance of the algorithms relative to each other are the number of arms, and the variance of the reward distribution. All other characteristics of the problem are shown to have less influence on relative performance of the algorithms.

Vast literature exists in general, we split our review over first more theoretical approaches, not necessarily taking context into account, and then the contextual case.

2.2 The classical Gittins index

With no context and independent conjugate priors, the discounted bandit decomposes into independent one-dimensional stopping problems. The resulting index policy is optimal and easy to compute J. Gittins 1974; J. Gittins, Glazebrook, and Weber 2011. The decomposition fails once context couples the arms.

2.3 Multi-armed bandit background

Krishnamurthy, Wahlberg, and Lingelbach 2005 use a value iteration approach to solve the multi-armed bandit problem using the Gittins index (see J. Gittins 1974, J. C. Gittins 1979 and J.

Gittins, Glazebrook, and Weber 2011 for more details) to compute the optimal strategy. The Gittins index gives an optimal strategy for the MAB problem with geometric time-discounted rewards. A downside of the approach is that it only works for an MAB problem without context. In Sebastian Pilarski, Slawomir Pilarski, and Varró 2021, the Bernoulli bandit problem is solved to optimallity through value iteration in a Bayesian manner, making use of a Beta-distribution as prior, which is similar to our approach. In that paper, limits on number of arms and time are practically needed to make the solution computationally tractable, and context is left out. Another relevant paper to consider is Granmo 2010, which also uses a Bayesian approach to solve the MAB, restricted to the two-arm bandit problem. A Beta-distribution is also used to model the Bernoulli bandit outcome, since it is a natural choice of distribution and has convenient properties for posterior calculation.

2.4 Contextual Bandit Background

The literature on the contextual case spans a diverse range of approaches with many contributions in recent years. A good starting point is Riquelme, Tucker, and Snoek 2018, which provides both an extensive overview of approaches to solve the contextual MAB problem, and a thorough benchmark of deep learning bandit algorithms. Many methods apply some form of either UCB or Thompson-Sampling to address the contextual MAB problem, and then focus on other interesting elements. For example, Schneider and Zimmert 2024 employ the full context when computing regret. In recent work, Xu, Min, and Wang 2024 and Kuroki et al. 2024 derive theoretical bounds for contextual Thompson Sampling by adapting the sampling procedure to the context and incorporating an additional optimistic bonus to enhance performance, a sort of combination of UCB and Thompson-Sampling. Other studies, such as Liu, Wei, and Zimmert 2024, consider adversarial bandits under linear assumptions, while Hanna, Yang, and Fragouli 2023 reduce the contextual bandit problem to a regular linear bandit to achieve competitive regret bounds. Meanwhile, Zhu and Rigotti 2021 propose an approach that bypasses complex posterior calculations by using frequentist sample mean and variance (so sample average uncertainty measures) in conjunction with deep learning to incorporate context. Duran-Martin, Kara, and Murphy 2022 combine Kalman filtering with a low-dimensional parameter subspace. Alternatively, Nilsson et al. 2024 uses tree ensembles instead of neural networks to incorporate context, but also uses UCB and Thompson-Sampling approaches to balance exploration and exploitation.

Together, these contributions demonstrate various ways of integrating contextual information and solving the MAB-problem. However, a practical approach that is able to take into account context, and at the same time solve the problem to optimality is missing.

3 Problem motivation: dynamic pricing of airline seats

We keep a single running example — the sale of paid seats on a flight — to link to throughout the paper. In this example, we have an airline which is considering prices for its seats on a specific flight. The airline has a range of prices it can consider, and after setting a price, it observes sales of the seats and with that the reward. For the airline, the challenge is in setting the right price that on the one hand optimizes its revenue by on the one hand maximizing the price it chooses, while at the same time not pricing the seat so high that it departs empty.

Context At each decision epoch t the airline observes a low-dimensional context c_t. Typical components are time-to-departure, day-of-week, holidays, type of flight, competitor prices, etc. We model the context as a stochastic process with Markov kernel $c_{t+1} \sim \mu_\xi(\cdot \mid c_t)$, where ξ is unknown (in Section 4 we will place a prior on it). This means that the context for the airline evolves, without the airline influencing the context directly. So we assume that the chocies made by the airline don't directly influence the evolution of the context.

Actions The airline quotes a price a_t chosen from a finite grid $\mathcal{A} = \{p_1, \ldots, p_K\}$; discretization is relatively standard in practice, and could e.g. also be up to the cent-level.

Rewards After setting price a_t, the airline sells a random number q_t of seats and earns revenue $r_t = a_t q_t$. We assume a Poisson demand model (although other distributions would work as well):

$$q_t \ \sim \ \text{Poisson}\big(\lambda_\theta(c_t, a_t)\big), \quad \lambda_\theta(c, a) := \exp\big(\theta^\top \phi(c, a)\big),$$

where $\phi(c, a)$ are fixed features (e.g. price, price^2, time-to-departure) and so are known elements, and θ is an unknown parameter vector shared across all prices that defines the relation between prices. Sharing this parameter means that information gathered at one price improves the demand estimate for every other price. Note that this also means that learning about the actions is not independent per action, as it is in a standard MAB problem.

Objective Over an undefined horizon we want to choose prices so as to maximize the expected γ-discounted revenue $\mathbb{E}\big[\sum_{t=0}^{T} \gamma^t \, a_t q_t\big]$, while learning both the demand parameter θ and the context dynamics ξ on-the-fly. This is precisely a contextual Bayesian bandit, and the next section translates the description above into the framework used in the rest of the paper.

4 A Bayesian framework for solving contextual MAB problems

We now formalize the model introduced in Section 3 and show how it results in a standard Markov decision process once we treat the agent's posterior as the state. In step 0 we give a high-level overview of where the algorithm is applied. Then we define the belief state and necessary constraints on this in step 1. Using those beliefs, we can compute the Bayesian updates in step 2. From this follows the reward of the action chosen in step 3. Taking those together, we can compute the optimal policy using value iteration, which is worked out in step 4.

4.1 Step 0: generative model

1. **Latent parameters.** At $t = 0$ we draw

$$\theta \sim p_0(\theta), \qquad \xi \sim p_0(\xi).$$

   In the seat-pricing example θ governs a single demand curve that is shared by every price, so learning at one price immediately updates the predicted demand for all other prices. The vector ξ parametrizes the context kernel μ_ξ, which defines the i context elements we have in our model. We define a prior p_0 that allows us to do this.

2. **Interaction loop** $t = 0, 1, 2, \ldots$

(a) Context c_t is revealed.

(b) Agent chooses price $a_t \in \mathcal{A}$.

(c) Seats sold: $q_t \sim \mathcal{D}_\theta(c_t, a_t)$ (Poisson with intensity $\lambda_\theta(c_t, a_t)$ in our example).

(d) Next context: $c_{t+1} \sim \mu_\xi(\cdot \mid c_t)$. So μ defines the distribution of the next context state. Note that that means we keep the context exogenous, and not dependent on the action chosen.

4.2 Step 1: representing beliefs

Before giving concrete examples we briefly justify the design constraints. We need to keep a finite posterior description, which is the most important constraint on the design. This is necessary, because value iteration needs a finite state coordinate. Conjugate families achieve that goal by construction; it works equally well to approximate the posterior if no analytical solution can be used, as long as the approximation is parameterized by finitely many real numbers.

Concrete choices used in our experiments

- **Demand parameter θ (shared across prices).** We assume a Poisson GLM $\lambda_\theta(c, a) = \exp(\theta^\top \phi(c, a))$ and place a Gaussian prior $\theta \sim \mathcal{N}(m_0, \Sigma_0)$. This is a logical choice because it allows analytical updates of the posterior, and an exponential shape is a standard choice for price elasticity.

- **Context kernel μ_ξ.** For a finite context set we choose the conjugate Dirichlet prior $\mu(\cdot \mid c) \sim \mathrm{Dir}(\alpha_{c1}^{(0)}, \ldots)$; in that case we can keep track of the state through the transition-count matrix $\alpha_{cc'}$, where $\alpha_{cc'}$ is the count of context transitions from state c to state c'.

Belief state All statistics collected at time t form $s_t = (m_t, \Sigma_t, a_t)$. Together with the current context c_t,

$$x_t = (s_t, c_t) \in \mathcal{X} = \mathcal{S} \times \mathcal{C}$$

constitutes the full MDP state.

4.3 Step 2: Computing the Bayesian update

Given x_t and the chosen action a_t, the agent knows the conditional laws of q_t and c_{t+1} — they depend only on the current posterior parameters. Once (q_t, c_{t+1}) are realized, the posterior parameters update via a closed-form rule. So

$$x_{t+1} = \mathcal{U}(x_t, a_t, q_t, c_{t+1})$$

can be computed in closed-form.

4.4 Step 3: reward

The immediate expected reward is revenue, $R(x_t, a) = a\, \mathbb{E}_{s_t}[\lambda_\theta(c_t, a)]$, bounded by $R_{\max} = p_{\max} q_{\max}$.

4.5 Step 4: Bellman operator and optimal policy

For any bounded $V : \mathcal{X} \to \mathbb{R}$

$$(TV)(x) = \max_{a \in \mathcal{A}} \{R(x, a) + \gamma\, \mathbb{E}[V(x') \mid x, a]\}.$$

Because T is a γ-contraction, value iteration converges to the unique fixed point V^* and the greedy action $a^*(x) = \arg\max_a (TV^*)(x)$ maximizes expected discounted revenue. Using the previous definitions and steps, we can apply value iteration to our contextual MAB problem, and with that find the optimal policy.

4.6 Step 5: exploitation/exploration

The greedy Bayes policy already weighs information gain against immediate revenue, since the value function combines the information gained from exploration through the posterior distribution with the expected revenue gain. Therefore there is no need to define an explicit exploration/exploitation policy, and is using this algorithm an optimal balance between exploration and exploitation.

4.7 Contextual Value Iteration algorithm

Taking the previous steps together, in its simplest form, the contextual value iteration algorithm can then be described as:

Algorithm 1 Contextual Value Iteration (finite-context variant)

```
Require: discount $\gamma$, horizon limit $H$, offline tolerance $\vartheta$
1: Offline: run value iteration on a grid of $(m, \Sigma, \alpha)$ until $\|V_{n+1} - V_n\|_\infty < \vartheta$; store $V^*$
2: for $t = 1, \ldots, H$ do
3:     observe context $c_t$ and posterior state $s_t$
4:     $a_t \leftarrow \arg\max_a \{R(x_t, a) + \gamma\, \mathbb{E}[V^*(x') \mid x_t, a]\}$
5:     play $a_t$, observe $q_t$ and $c_{t+1}$
6:     update $(m, \Sigma)$ via posterior calculation
7:     $\alpha_{c_t, c_{t+1}} \leftarrow \alpha_{c_t, c_{t+1}} + 1$
8: end for
```

5 Experimental Evaluation

In this section we we practically show the performance of the value iteration approach. In 5.1 we describe both the algorithm and the performance in the case without context, then in 5.2 we do so for the contextual case in a toy example, and in 5.3 we then move to a full airline pricing model.

5.1 Value Iteration for Bernoulli Bandits

First we experimentally test the results without context, with a Bernoulli bandit. In this specific setting, we define the state as a Beta distribution describing our belief about the p-value of the Bernoulli distribution of this arm. So we define the state $s = \{(\alpha_1, \beta_1), (\alpha_2, \beta_2), \ldots, (\alpha_k, \beta_k)\}$ where for each arm i, (α_i, β_i) are the parameters of the Beta distribution representing the belief over its

Bernoulli payoff. This makes updating the posterior distribution very simple, simply incrementing either the α or β parameter, dependent on a win or loss. The action space is defined as $A = \{1, 2, \ldots, k\}$, where choosing action a corresponds to pulling arm a. Note that the immediate reward is defined as 1 for a success or 0 for a failure, which is why the win branch in the value update includes an additive term of 1. Pseudo-code for the approach is in algorithm 2.

Algorithm 2 Value Iteration for Bernoulli Bandit problem

```
Require: • $\theta$ and $max\_iterations$, pre-defined convergence criteria
 1: Initialize: $value\_function \leftarrow zeros(|StateSpace|)$
 2: $\Delta \leftarrow 0.0$
 3: $iteration \leftarrow 0$
 4: while $iteration < max\_iterations$ and $\Delta > \theta$ do
 5:     $\Delta \leftarrow 0.0$
 6:     for all state $s \in StateSpace$ do
 7:         $current\_value \leftarrow value\_function[s]$
 8:         $max\_value \leftarrow -\infty$
 9:         for all action $a \in \{1, 2, \ldots, num\_bandits\}$ do
10:             $cur\_alpha \leftarrow \alpha$ value of arm $a$ in state $s$
11:             $cur\_beta \leftarrow \beta$ value of arm $a$ in state $s$
12:             $win\_prob \leftarrow \frac{cur\_alpha}{cur\_alpha + cur\_beta}$
13:             $win\_state \leftarrow s$ with $cur\_alpha$ incremented by 1
14:             $lose\_state \leftarrow s$ with $cur\_beta$ incremented by 1
15:             $win\_val \leftarrow value\_function[win\_state]$
16:             $lose\_val \leftarrow value\_function[lose\_state]$
17:             $action\_value \leftarrow win\_prob \times (1 + \gamma \times win\_val) + (1 - win\_prob) \times (\gamma \times lose\_val)$
18:             if $action\_value > max\_value$ then
19:                 $max\_value \leftarrow action\_value$
20:             end if
21:         end for
22:         $value\_function[s] \leftarrow max\_value$
23:         $\Delta \leftarrow \max(\Delta, |current\_value - max\_value|)$
24:     end for
25:     $iteration \leftarrow iteration + 1$
26: end while
27: for all state $s \in StateSpace$ do
28:     $optimal\_policy[s] \leftarrow \arg\max_a\{value\_function(s, a)\}$
29: end for
30: return $value\_function, optimal\_policy$
```

A computational advantage of this approach is that the value function can be calculated offline, while during inference, only a look-up in the value function has to be done. This means that during inference, the approach requires no computation. Calculating the value function could, however, be computationally expensive. For each arm, every possible combination of the distribution parameters would have to be computed. This implies that the total number of possible combinations could potentially be very large as for every arm, every possible combination of α and β would have to be computed and those parameters are bound by the number of rounds H leading to $(H \cdot H)^k$ states that need to be computed and stored. With a relevant number of rounds, this grows very quickly in k. But there are various optimizations to consider that address this curse of dimensionality. First, we can observe that as long as we have no prior information that one arm performs better than the others, we should assume an equal prior over all arms. Then, if all arms belong to a Bernoulli bandit and are independent from each other, the value function itself is the same for every separate arm. This implies the value function only depends on α and β. So instead of calculating and storing the value function for each arm, we only have to do this for every possible combination of α and β offline. We can then use the outcome of this calculation for every individual arm - so in the online calculation we have a state space of $k(H + H)$, with a value for α and β per arm. We then do a look-up based on these parameters in the value function calculated offline. In this way, we can easily compute the full value function.

A problem is then still that with an infinite time horizon H, either α or β can grow to infinity and the value function can not be calculated completely. To approximate the value function, we set a (possibly large) time limit which bounds H, and with that the state space of α and β. If H is set to some value h, the state space for the value function to be computed is $(h + 1)^2$. In the case of Bernoulli bandits, a logical prior is the Beta-distribution, which gives an analytical posterior which is very easy to compute, so the computational load is limited. Note that the algorithm can easily be extended to other types of distributions as well, as long as it's possible to efficiently calculate a posterior. Besides, setting this maximum value for h is also directly an advantage of the algorithm. In practice there are many applications where we don't have an infinite time window, and there is a maximum number of rounds.

Another advantage of this algorithm is that by selecting a value for γ, the importance of immediate vs long-term rewards can be set, which can direct how much to explore vs exploit. This is not something that is easy to do through algorithms like Thompson Sampling, UCB or epsilon-greedy approaches. It is implicitly possible by setting a higher/lower value for ϵ or c to promote more exploration, but this does not translate easily to the right value for these. However, this also translates into a downside, as γ must be smaller than 1, meaning a certain discount factor is required for the algorithm to work.

5.1.1 Experimental Setup

- **Bandit Environment:** We consider a Bernoulli bandit with $k = 10$ and 100 arms. For each arm i, a base probability p_i is sampled uniformly at random from $[0, 1]$. Rewards are binary (0 or 1). The time horizon H was set to values $50, 100, 200, 500, 1000$.

- **Algorithms Compared:**

  1. **Value Iteration:** Our Value Iteration algorithm described in algorithm 2.
  2. **Epsilon greedy**, a standard approach with $\epsilon = 0.05$.
  3. **Epsilon decreasing** where we gradually decrease the epsilon value with a factor α, set at 0.98.
  4. **Upper Confidence Bound**, the standard UCB algorithm.
  5. **Thompson Sampling**, the standard Thompson Sampling algorithm.

- **Simulation Parameters:** Each algorithm is run for H rounds per experiment, and each configuration is repeated 1000 times. Performance is measured by the average reward per round.

5.1.2 Performance of the Value Iteration algorithm without context

In figures 1 and 2, the total discounted reward is displayed for the various algorithms with an increasing number of rounds in a 10 and 100 bandit setting, respectively. Instead of displaying the total discounted reward, the reward is shown relative to the UCB algorithm (with UCB performance = 1). It is interesting to see that the value iteration approach is generally able to learn faster than the benchmark algorithms, and has a better trade-off between exploration and exploitation. Only in settings where there is more time to learn (so with a low number of arms, and a high number of rounds), do algorithms like Thompson Sampling or epsilon-greedy approach value iteration performance. And conversely, especially in a setting with a relatively high number of arms and a relatively low number of rounds, value iteration is significantly better able to capture early benefits and with that have a better total performance over time.

5.1.3 Computation times

Calculating the value function offline for a problem instance with a maximum of 300 rounds, takes about 60 seconds on a regular 2021 Macbook Pro (note that the calculation time does not depend on the number of arms). Then running the value iteration algorithm online a 1000 times on a problem instance with 100 arms, for 300 rounds, takes about 30 seconds. Memory usage is also no issue due to the way the state space and value function are stored. With that, the algorithms are easily tractable on any modern hardware.

5.2 Contextual value iteration

In this section, we benchmark our Contextual Value Iteration (CVI) algorithm against two baselines: Contextual Epsilon-Greedy (CEG) and Contextual Thompson Sampling (CTS) in the contextual Bernoulli bandit setting. Our goal is to solve the contextual MAB problem, where the reward distribution of each arm is influenced non-linearly by the context.

5.2.1 Experimental Setup

- **Bandit Environment:** We consider a contextual Bernoulli bandit with 5 arms. For each arm i, a base probability p_i is sampled uniformly at random from $[0, 1]$. Rewards are binary (0 or 1). The context is a discrete variable, randomly sampled from the possible values:

$$\mathcal{C} = \{0, 1, 2, 3, 4\}.$$

- **Context Function:** We use the following non-linear context function:

$$\texttt{context\_function}(i, c) = \begin{cases} \min\bigl(2 \cdot p_i,\, 1\bigr), & \text{if } c = i, \\ 0.75 \cdot p_i, & \text{otherwise,} \end{cases}$$

  which doubles the base probability when the context matches the arm index, and scales it by 0.75 otherwise.

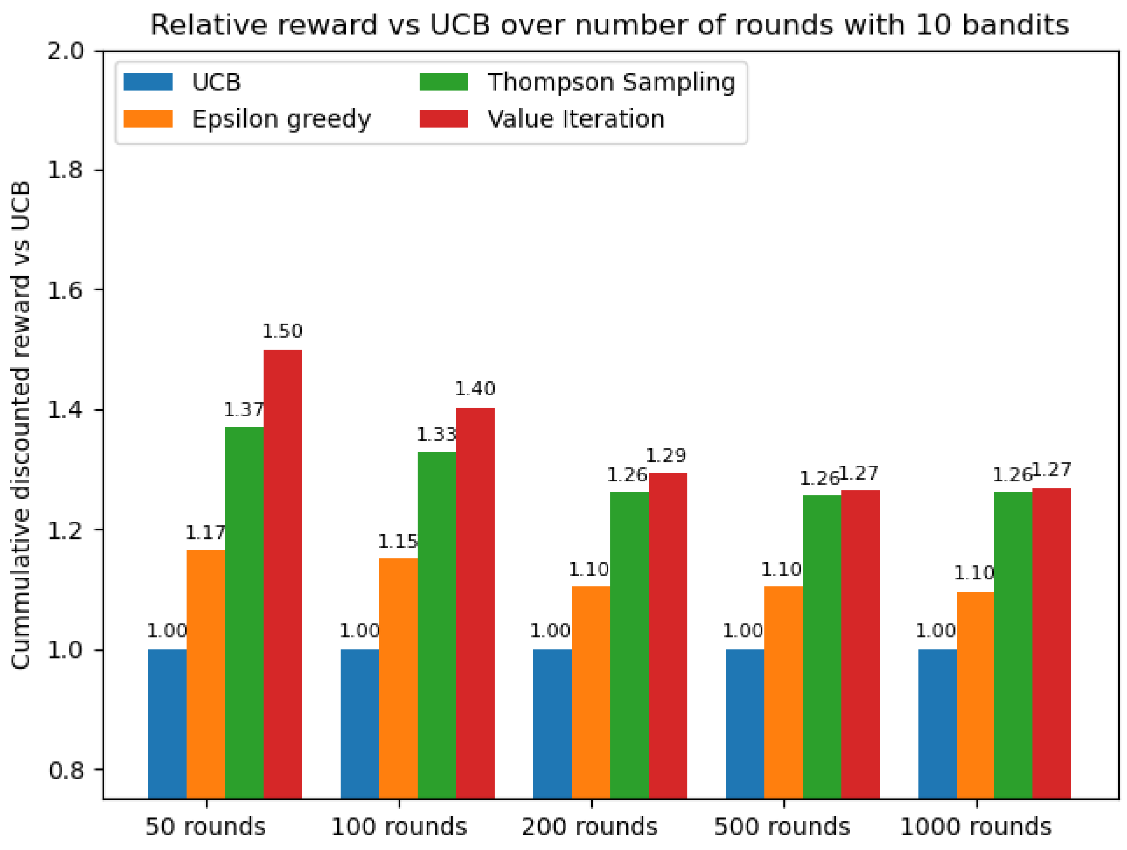

Figure 1: Relative performance (total discounted reward normalized by UCB performance) for 10 bandits setting over varying time horizons $H = 50, 100, 200, 500, 1000$

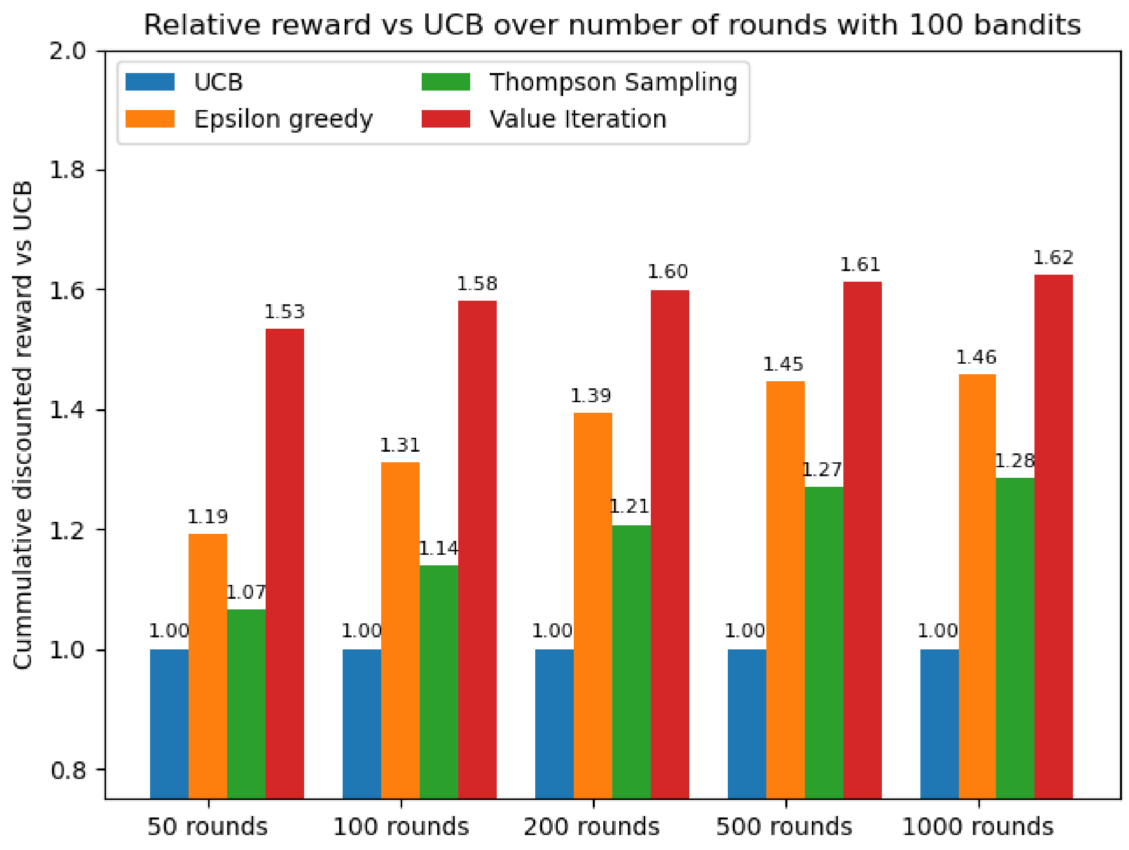

Figure 2: Relative performance (total discounted reward normalized by UCB performance) for 100 bandits setting over varying time horizons $H = 50, 100, 200, 500, 1000$

- **Algorithms Compared:**
    1. **CVI:** In CVI, each arm's state is defined as (α, β, c) and the value function $V(\alpha, \beta, c)$ is computed offline over a three-dimensional grid. Some implementation details are described in 5.2.2.
    2. **Contextual Epsilon-Greedy (CEG):** An extension of epsilon-greedy that maintains separate win and pull counts per arm and per discretized context.
    3. **Contextual Thompson Sampling (CTS):** A method that maintains independent Beta parameters for each arm in each context and selects arms by sampling from the corresponding Beta distributions.
- **Simulation Parameters:** Each algorithm is run for 250 rounds per experiment, and each configuration is repeated 1000 times. Performance is measured by the average reward per round.

5.2.2 Implementation Details and Trade-Offs for CVI

For CVI, our implementation involves the following key components:

1. **State Augmentation:** Each arm maintains separate Beta parameters for each discretized context, resulting in a state (α, β, c).
2. **Initial Offline Value Function Computation:** Using the prior, the value function is computed offline over the full state grid.
3. **Online Update of Value Function:** As the algorithm interacts with the environment, a binary mask tracks which states have been visited. So there is a 0/1 database stored that indicates which states have been visited. Every 25 rounds, we update the value function for these states (and any states dominated by them) to reflect the new posterior information that is gained from the context that is now available online. This is done in this manner to speed up online calculation. Note that this is necessary and can't be calculated offline, since we don't put any assumption offline on how the context influences the pay-off.
4. **Learning Context Transitions:** The algorithm also updates an online estimate of the context transition matrix based on observed context changes.

These design choices balance computational cost with the need for the value function to adapt as more data is collected. Note that this is not the exact same calculation as described earlier, but balances some computational needs. We could also run the algorithm exactly as it was described earlier, but at greater computational cost.

5.2.3 Results

Figure 3 shows the simulated performance. The experimental results indicate that:

- CVI is well able to collect data and perform periodic planning updates, in which the performance quickly improves and outperforms other algoritms.
- The empirical average reward per round increases over time, consistent with our theoretical approximation, although the practical performance far exceeds the worst-case bound.

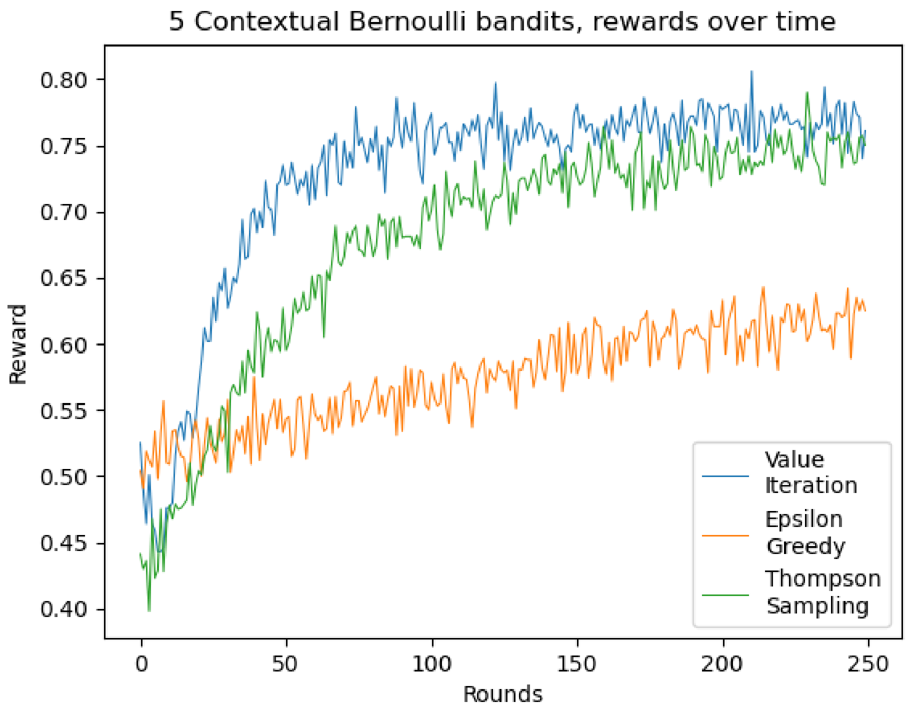

Figure 3: Average reward per round for CVI, CEG, and CTS over 250 rounds, averaged over 1000 runs.

5.3 Contextual Pricing with Poisson Demand

We now apply Contextual Value Iteration (CVI) to a small but complete pricing problem. The model is in line with standard revenue management practice while remaining computationally tractable so that we can compute a discretized V^* offline and execute online with table look-ups only.

Model At each round t we observe a discrete context $c_t \in \{0,1\}$, with $c=0$ denoting leisure and $c=1$ business. We choose a price $a_t \in \mathcal{P}$ from a small finite grid and realize seats sold

$$y_t \sim \text{Poisson}(\lambda_\theta(c_t, a_t)), \qquad \lambda_\theta(c,a) \;=\; \exp\bigl(\theta^\top \phi(c,a)\bigr),$$

with revenue $r_t = a_t\, y_t$. Context evolves exogenously via a Markov kernel $\mu(c' \mid c)$.[1]

To capture that business is less price elastic, we use the 4-dimensional feature map

$$\phi(c,a) \;=\; \bigl[\,1,\; a,\; \mathbf{1}\{c{=}1\},\; \mathbf{1}\{c{=}1\}\cdot a\,\bigr]^\top,$$

with corresponding parameter names $\theta = \bigl[\theta_0,\, \theta_p,\, \theta_{b0},\, \theta_{bp}\bigr]^t$. This gives the log-intensity $\log \lambda_\theta(c,a) = \theta_0 + \theta_p \cdot a + \theta_{b0}\mathbf{1}\{c{=}1\} + \theta_{bp}\mathbf{1}\{c{=}1\} \cdot a$ so that leisure has log–intensity $\theta_0 + \theta_p a$ and business $\theta_0 + \theta_{b0} + (\theta_p + \theta_{bp})a$.

Belief and planning We maintain a Gaussian belief on θ: $\theta \sim \mathcal{N}(m, \Sigma)$ and update the mean with a one-step Laplace/EKF correction,

$$m' \;=\; m + \underbrace{\frac{\Sigma x}{1 + \lambda\, x^\top \Sigma x}}_{\text{Sherman–Morrison}} \bigl(y - \lambda\bigr), \quad \lambda = \exp(x^\top m), \;\; x = \phi(c,a),$$

while keeping Σ fixed to the prior covariance Σ_0 to keep the belief state finite. We discretize m on a tensor grid $\mathcal{M}$ centered at the prior mean and define the belief state as $(m,c) \in \mathcal{M} \times \{0,1\}$. The Bellman update at (m,c) takes the form

$$(TV)(m,c) \;=\; \max_{a \in \mathcal{P}} \Bigl\{ a\,\lambda \;+\; \gamma\, \mathbb{E}_{y \sim \text{Pois}(\lambda)} \bigl[\mathbb{E}_{c' \sim \mu(\cdot|c)} V(m'(y), c') \bigr] \Bigr\},$$

with the Poisson expectation truncated at $y_{\max}$ (tail mass lumped at $y_{\max}$). Running value iteration on this finite state space yields a table V^* and a greedy policy $\pi^*(m,c)$ that we use online by look-up.

Experimental setup True parameters are chosen to yield realistic elasticities:

$$\theta_p = -0.013, \;\; \theta_{bp} = +0.009, \;\; \theta_{b0} = +0.15,$$

and θ_0 is calibrated so that leisure has $\mathbb{E}[y \mid a{=}100] \approx 8$. The agent's prior is close but miscalibrated on the slopes to create room for learning:

$$m_0 \;=\; \theta_{\text{true}} + (-0.10,\; +0.008,\; -0.02,\; -0.006), \quad \Sigma_0 = \text{diag}(0.30^2,\; 0.025^2,\; 0.20^2,\; 0.025^2).$$

[1] In the experiment we use $\mathcal{P} = \{60, 80, 100, 120\}$ (currency units), two contexts, and the transition matrix $\mu = \left(\begin{smallmatrix} 0.92 & 0.08 \\ 0.06 & 0.94 \end{smallmatrix}\right)$.

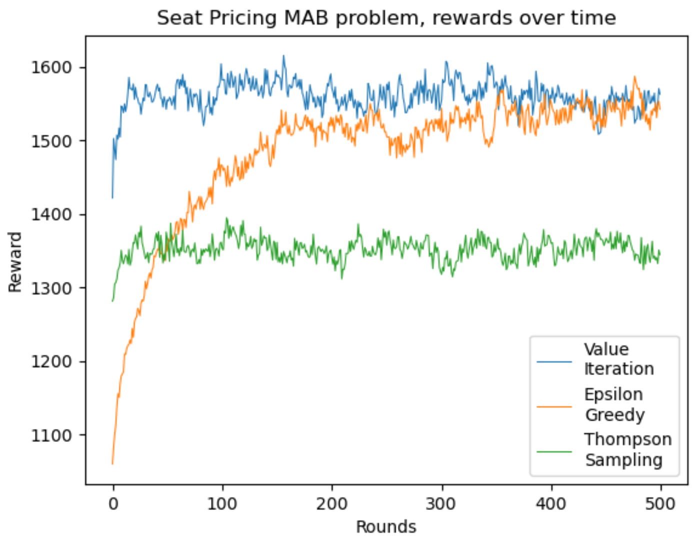

Figure 4: Seat pricing (Poisson GLM) — average revenue per round over 1000 episodes, 250 rounds each. CVI uses an offline V^* on a small belief grid and greedy look-up online; baselines are contextual ε-greedy and contextual Thompson sampling.

We grid each coordinate of m with 5 points over ± 1.5 prior standard deviations (a $5\times5\times5\times5$ grid), set $\gamma = 0.99$, truncate the Poisson support at $y_{\max} = 12$, and jitter the episode start belief as $m_0 + \epsilon$ with $\epsilon \sim \mathcal{N}(0, (0.5^2)\,\mathrm{diag}(\Sigma_0))$ to avoid identical early actions across episodes. We compare against contextual ε-greedy and contextual Thompson sampling using the same GLM and features. Each method is run for 1000 episodes of 250 rounds; we report per-round averages.

Results Figure 4 plots the average revenue per round. CVI starts below its eventual plateau and improves during the first tens of rounds before stabilizing above the heuristics. Learning is present but modest, which is logical in this toy regime with (i) a small action set, (ii) only two contexts, and (iii) a prior already close to truth. Wall-clock on an M1 laptop: computing V^* once takes $\approx$170 seconds; the subsequent online phase (all 1000×250 decisions) takes $\approx$34 seconds versus $\approx$24 seconds for ε-greedy and $\approx$92 seconds for Thompson sampling.[2]

Takeaway This experiment demonstrates that CVI extends cleanly from the contextual Bernoulli setting to a pricing problem with Poisson demand and learning across prices: with a small, fully offline discretization the method is practical, yields higher revenue than standard heuristics, and preserves the analytical structure needed for deployment. The toy nature limits the observed learning signal; richer feature maps, larger price grids, or slightly less informative priors increase learning without changing the mechanics.

[2]The offline cost is amortized across episodes; once V^* is available, per-round execution is a table look-up plus a closed-form mean update.

5.4 Discussion

The experimental results show that our CVI algorithm is capable of adapting to the non-linear influence of context on reward distributions, both in a purely toy simulation, and in a more realistic pricing simulation. By periodically updating the value function in visited regions of the state space, CVI successfully uses both the offline computation based on the prior, and the online posterior updates, while balancing computational needs. While CVI incurs a higher computational cost compared to CEG and CTS (in this case, approximately 140 seconds for 1,000 simulations of 250 rounds in our simulation, compared to 3 seconds for Epsilon Greedy and 20 seconds for Thompson Sampling), its ability to achieve higher average rewards demonstrates its practical viability in such settings. However, due to the curse of dimenstionality, it is infeasible to run the CVI algorithm on larger states with more complex contexts, arms and reward fucntions. Future work should consider leveraging function approximation techniques, such as deep learning, to mitigate this problem.

6 Conclusion

By defining the state of a contextual MAB problem completely as its posterior estimates, the entire problem becomes an MDP, and we can use standard tools like value iteration to optimally solve the contextual multi-armed bandit problem. Earlier known optimal solutions for standard bandit problem like the Gittins index, could be seen as a specific case of this wider definition used in this paper. Therefore it is not necessary to use heuristics to balance exploration and exploitation, we can also use the mathematically optimal balance. We have shown that our approach also works on practical contextual MAB instances, and that the approach works well and can be adapted to different situations.

This approach only applies if a logical prior can be chosen. Without a good choice of priors, the entire approach breaks down. This ia also a challenge in the evaluation of the performance of the algorithm, starting with exactly the right prior, means that the algorithm is optimal from the start. While starting from a prior that doesn't cover the true distribution means that the optimal solution can never be found. Besides, this approach is impacted by the curse of dimensionality, and larger problem sizes become computationally infeasible. Further research should be directed towards real-world large problem instances, where this fully analytical approach would break down due to this curse of dimensionality. For instance using function approximation techniques like deep learning, the value function could be approximated, and arbitrarily large problem sizes could also be handled. Furthermore, we now require a stationary context that is independent from the action chosen. It would also be interesting to investigate how more flexibility could be added there.

References

Thompson, William R (1933). "On the likelihood that one unknown probability exceeds another in view of the evidence of two samples". In: *Biometrika* 25.3-4, pp. 285–294.

Robbins, Herbert (1952). "Some aspects of the sequential design of experiments". In.

Gittins, John (1974). "A dynamic allocation index for the sequential design of experiments". In: *Progress in statistics*, pp. 241–266.

Gittins, John C (1979). "Bandit processes and dynamic allocation indices". In: *Journal of the Royal Statistical Society Series B: Statistical Methodology* 41.2, pp. 148–164.

Lai, Tze Leung and Herbert Robbins (1985). “Asymptotically efficient adaptive allocation rules”. In: *Advances in applied mathematics* 6.1, pp. 4–22.

Cesa-Bianchi, Nicolo and Paul Fischer (1998). “Finite-time regret bounds for the multiarmed bandit problem.” In: *ICML*. Vol. 98. Citeseer, pp. 100–108.

Auer, Peter, Nicolo Cesa-Bianchi, and Paul Fischer (2002). “Finite-time analysis of the multiarmed bandit problem”. In: *Machine learning* 47, pp. 235–256.

Krishnamurthy, Vikram, Bo Wahlberg, and Frank Lingelbach (2005). “A value iteration algorithm for partially observed markov decision process multi-armed bandits”. In: *Math. of Oper. Res*, pp. 133–152.

Granmo, Ole-Christoffer (2010). “Solving two-armed Bernoulli bandit problems using a Bayesian learning automaton”. In: *International Journal of Intelligent Computing and Cybernetics* 3.2, pp. 207–234.

Gittins, John, Kevin Glazebrook, and Richard Weber (2011). *Multi-armed bandit allocation indices*. John Wiley & Sons.

Kuleshov, Volodymyr and Doina Precup (2014). “Algorithms for multi-armed bandit problems”. In: *arXiv preprint arXiv:1402.6028*.

Riquelme, Carlos, George Tucker, and Jasper Snoek (2018). “Deep bayesian bandits showdown: An empirical comparison of bayesian deep networks for thompson sampling”. In: *arXiv preprint arXiv:1802.09127*.

Pilarski, Sebastian, Slawomir Pilarski, and Dániel Varró (2021). “Optimal policy for Bernoulli bandits: Computation and algorithm gauge”. In: *IEEE Transactions on Artificial Intelligence* 2.1, pp. 2–17.

Zhu, Rong and Mattia Rigotti (2021). “Deep bandits show-off: Simple and efficient exploration with deep networks”. In: *Advances in Neural Information Processing Systems* 34, pp. 17592–17603.

Duran-Martin, Gerardo, Aleyna Kara, and Kevin Murphy (2022). “Efficient online bayesian inference for neural bandits”. In: *International Conference on Artificial Intelligence and Statistics*. PMLR, pp. 6002–6021.

Hanna, Osama A, Lin Yang, and Christina Fragouli (2023). “Contexts can be cheap: Solving stochastic contextual bandits with linear bandit algorithms”. In: *The Thirty Sixth Annual Conference on Learning Theory*. PMLR, pp. 1791–1821.

Kuroki, Yuko et al. (2024). “Best-of-Both-Worlds Algorithms for Linear Contextual Bandits”. In: *International Conference on Artificial Intelligence and Statistics*. PMLR, pp. 1216–1224.

Liu, Haolin, Chen-Yu Wei, and Julian Zimmert (2024). “Bypassing the simulator: Near-optimal adversarial linear contextual bandits”. In: *Advances in Neural Information Processing Systems* 36.

Nilsson, Hannes et al. (2024). “Tree ensembles for contextual bandits”. In: *arXiv preprint arXiv:2402.06963*.

Schneider, Jon and Julian Zimmert (2024). “Optimal cross-learning for contextual bandits with unknown context distributions”. In: *Advances in Neural Information Processing Systems* 36.

Xu, Ruitu, Yifei Min, and Tianhao Wang (2024). “Noise-adaptive thompson sampling for linear contextual bandits”. In: *Advances in Neural Information Processing Systems* 36.