Task 2 report

Author: Konstantin Fedorov B21-DS02, k.fedorov@innopolis.university

Based on tutorial: `https://web.stanford.edu/~bvr/pubs/TS_Tutorial.pdf`.
In given tutorial different approaches of online learning were described: *Greedy, Upper Confidence Bound (UCB) and Thompson sampling.* To better understand them let's consider multi-arm Bandit problem.

Multi Arm Bandit

In this problem we have **n** different actions or arms that we can use in each step. Every action give some stochastic reward. Our goal is to maximize reward sum with limited number of actions.

Bernoulli Bandit

This is a Multi Arm Bandit problem. When action k played, reward of one is produced with probability θ_k and reward of zero with probablity $1 - \theta_k$. All θ are unknown, but fixed over time. Hence, reward is Bernoulli variable. For this problem Bayesian inference with Beta as prior distribution works very well. $\theta \sim Beta(\alpha, \beta)$ where we can interpret α as number of success + 1, β is number of failures + 1. Our prior assumption is that every θ possible, therefore $Uniform(0, 1)$, which can be expressed with $Beta(1, 1)$. Updating parameters can be done in such manner:

$$\alpha = \alpha + reward$$
$$\beta = \beta + 1 - reward$$

Mean or expected value of θ:

$$E(\theta) = \frac{\alpha}{\alpha + \beta}$$

Greedy

Every iteration, algorithms computes expected value of θ_k for every action k and chooses action with highest probability. Then it observes result and update α_k, β_k. This approach always use action with biggest expected value of probability. Therefore, it explores badly and often converges to local optima.

UCB

Upper Confidence Bound approach computes following expression for every action k:

$$U_t(k) = \mu_k + c\sqrt{\frac{1.5 \cdot ln(t)}{N_t(k)}}$$

With our prior as $Beta$, we can compute:

$$\mu_k = \frac{\alpha_k}{\alpha_k + \beta_k}$$
$$N_t(k) = \alpha_k + \beta_k$$

c is the hyperparameter *degree of optimism.* If $c = 1$, UCB is called UCB1.
Then chooses action k with the biggest value, observes result and update α_k, β_k.

Thompson sampling

In general Thompson sampling can be described as following process:

- Sample reward θ_k from priors for every action k
- Select biggest reward, observe result
- Update prior distribution based on result

For Bernoulli reward we use Beta distribution as prior and update it accordingly. This simple approach gives us good performance without need of tuning additional hyperparameters.

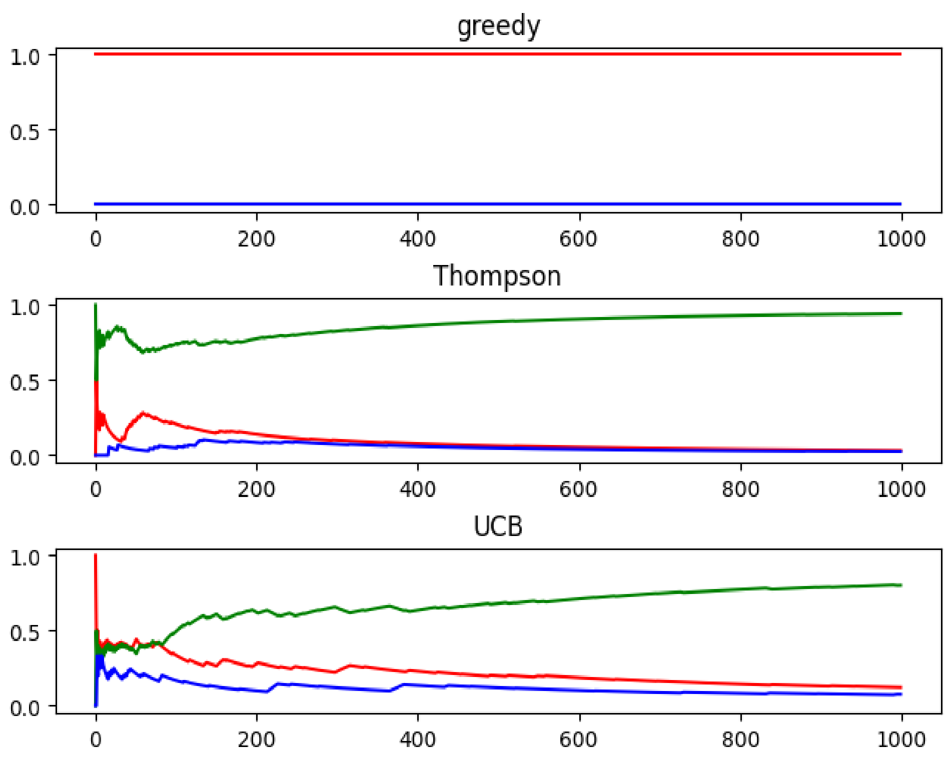

Figure 1: X axis is iteration. Y axis is portion of total number of actions that were choosen until this iteration. Red is 1 action (0.7 probability), Blue is (0.8 probabity)

Plot and conclusion

Let's test this algorithms on Bernoulli Bandit problem. There would be 3 arms with $\theta = [0.7, 0.8, 0.9]$, number of iterations is 1000.

From plots Figure (1,2) we can see that Thompson sampling work well with UCB1 slightly behind, while Greedy is stuck in local optima.

Gaussian Bandit

It is Multi Arm Bandit problem where every action k reward is sampled from normal distribution with predefined μ_k and σ_k. To solve this problem let's consider UCB and Thompson methods.

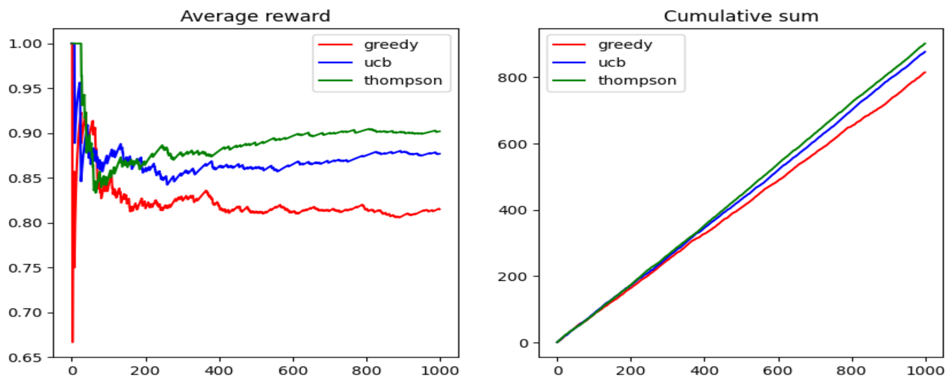

Figure 2: X axis in iteration, Y axis average reward from begining until this iteration

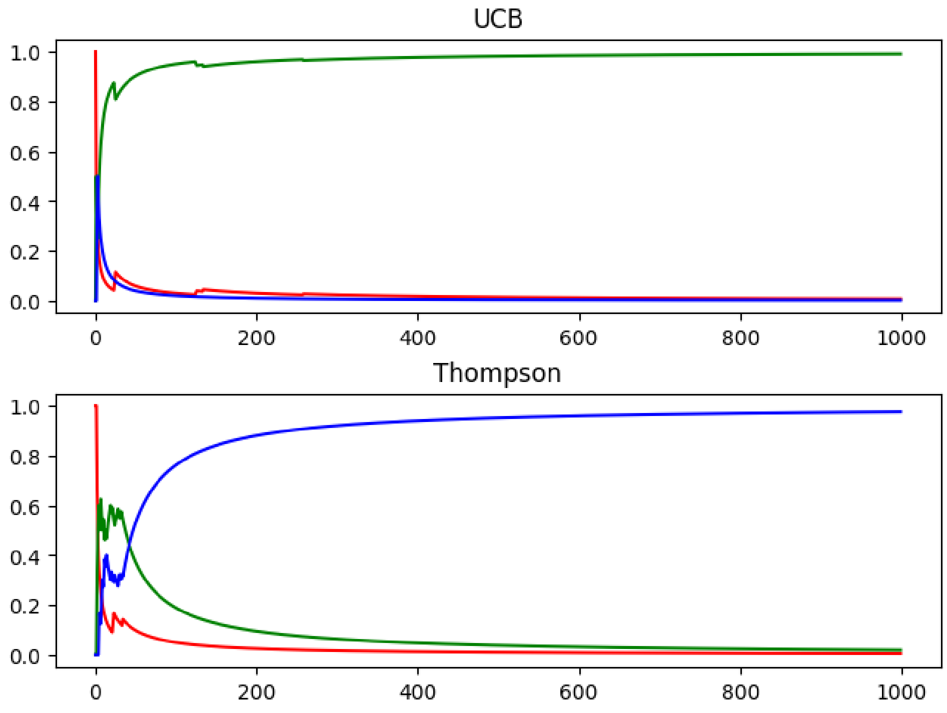

Figure 3: X axis is iteration. Y axis is portion of total number of actions that were choosen until this iteration. Red is 1 action (0.7 probability), Blue is (0.8 probabity)

Gaussian UCB

Implementation of UCB is similar to Bernoulli Bandit, however μ will be computed as average reward and $N_k(t)$ is just count of how many times action was called.

Gaussian Thompson sampling

In this approach we use normal prior to estimate reward. Initial parameters are assumed as $\mu_0 = 0,\ \sigma_0 = 10000$. In order to update we use conjugate properties of normal distribution with known variance σ^2:

$$\sigma_0 = \left(\frac{1}{\sigma_0^2 + \frac{n}{\sigma^2}}\right)^{-1}$$

$$\mu_0 = \frac{1}{\frac{1}{\sigma_0^2} + \frac{n}{\sigma^2}} \left(\frac{\mu_0}{\sigma_0^2} + \frac{\sum_{i=1}^{t} x_i}{\sigma^2}\right)$$

Where t is how many times action was called, $\sum_{i=1}^{t} x_i$ is sum all of observerd action's reward.

Plots and conlusion

Algorithm are tested with arm's $\mu = [1, 2, 3]$ and $\sigma = [1, 2, 3]$ with 1000 iterations. From Figures (3,4) we can see that Thompson sampling performes well. While conducting this test multiple times, UCB was close and even outperformed Thompson sampling in some cases. However, in number of trials UCB stucks to not optimal action due to high variance of data.

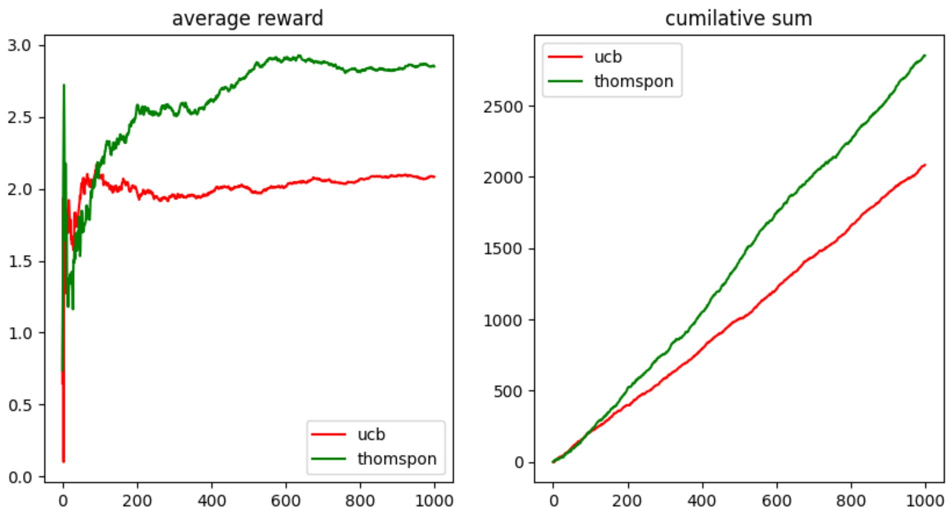

Figure 4: X axis in iteration

Product Assortment

In this problem agent ample supply of n types of products. The seller gains profit p_i per unit sold of product. The agent can offer only k products to customers. Demand of product depends on which items were offered with this product. In order to maximize profit, the seller should carefully select optimal set of products. We represent agent's decision as vector $x \in \{0,1\}^n$ where $x_i = 1$ indicates that product is offered and $x_i = 0$ indicates otherwise. Upon offering an assortment, the agent observes random log-Gaussian-distributed demand d. The mean of distribution depends on entire assortment x and not known matrix $\theta \in R^{n\times n}$.

$$\ln(d_i)|\theta, x \sim N((\theta x)_i, \sigma^2)$$

Where σ^2 is known variance of demand. To compute total profit and it's expected value:

$$E\left[\sum_{i=1}^{n} p_i x_i d_i | \theta, x\right] = \sum_{i=1}^{n} p_i x_i e^{(\theta x)_i + \frac{\sigma^2}{2}}$$

As our prior for θ we can use Multivariate Normal distribution with $\mu \in R^{n^2}$, $\Sigma \in R^{n^2\times n^2}$, because θ has n^2 elements.

Let's introduce $z \in R^k$ and defined as $z = \ln(\hat{d})$, where $\hat{d}$ is demand of selected products.
Let S be a $k \times n$ "selection matrix", where $S_{j,ind_j} = 1$ and $ind_j \in R^k$ contains index of selected in assortment product. Other values of S are zero. Also define:

$$W = x^T \otimes S,\ W \in R^{k\times n^2}$$

We can update our μ, Σ as follows:

$$\mu \leftarrow \left(\Sigma^{-1} + \frac{1}{\sigma^2}W^T W\right)^{-1}\left(\Sigma^{-1}\mu + \frac{1}{\sigma^2}W^T z\right)$$
$$\Sigma \leftarrow \left(\Sigma^{-1} + \frac{1}{\sigma^2}W^T W\right)^{-1}$$

As our test environment we use $n = 6$ and $\sigma = 0.4$, $p_i = 1/6$. As initial prior parameters $\mu = 0$ and Σ has variance of 1 on diagonal and 0.2 otherwise. Let's compare Greedy and Thompson methods. Greedy algorithm choose action with highest expected value of profit and updates parameters, while Thompson Sampling will sample profit from distribution.

From Figure 5 we can see that Thompson algorithm performes slightly better than Greedy. This is contrary to tutorial plot, where Thompson sampling significantly outperforms Greedy algorithm.

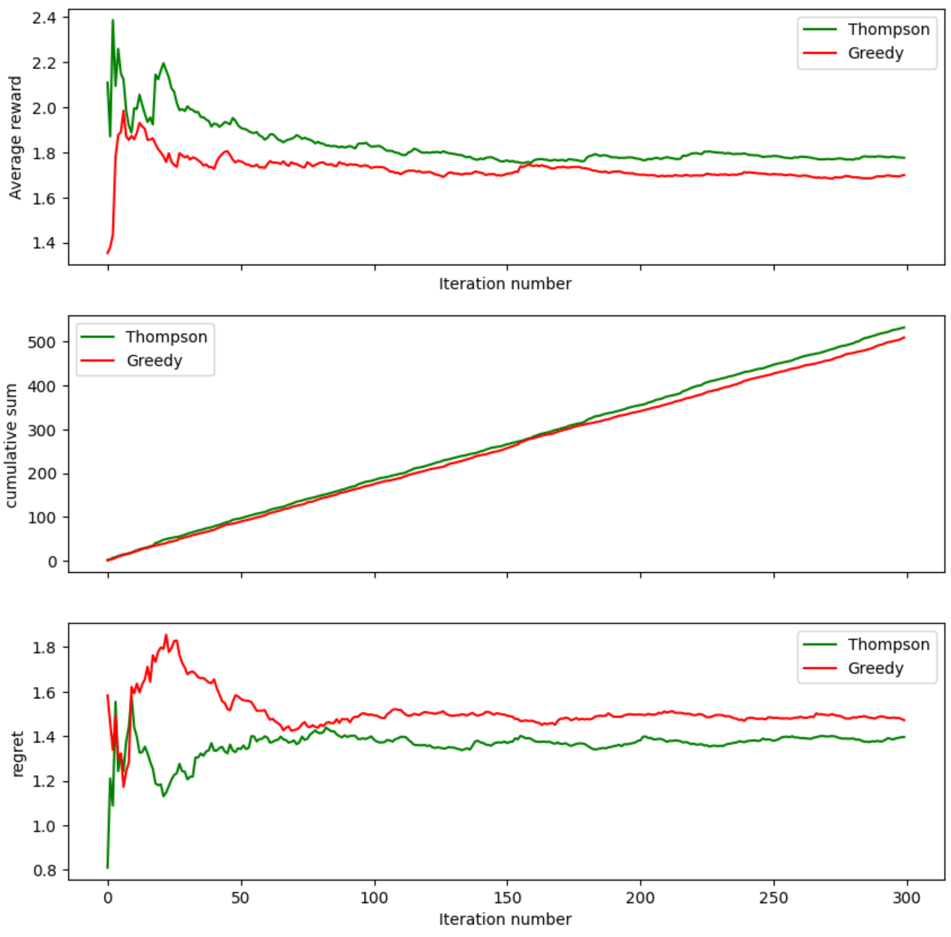

Figure 5: X axis in iteration