Contextual Linear Bandits in Recommender Systems

Alireza Nouri
Sharif University of Technology
Teymoori Sq, Tehran, Iran
alireza.nouri77@sharif.edu

Mir-Omid Haji-Mirsadeghi
Sharif University of Technology
Teymoori Sq, Tehran, Iran
mirsadeghi@sharif.edu

Abstract

Probabilistic reinforcement learning is a powerful approach for modeling the distribution of arms in recommender systems. In this paper, we explore two methods for optimizing advertisement display to maximize cumulative rewards. The first method involves online logistic regression to identify parameters that maximize the probability of a click for each ad displayed. The second method utilizes Thompson Sampling with Laplace approximation to learn the distribution of arms and select the ads most likely to receive clicks. We implement these algorithms on simulated data and compare their performance.

1 Introduction

Sequential learning is a fundamental concept in machine learning and also plays an important role in recommender systems. In this scope, we do not access all data at first, and each data point arrives at each time step. One of the common algorithms used in this concept is logistic regression to map data between 0 and 1. In this concept, logistic regression is refined when new data arrives [1,2]. Thompson sampling is one of the most popular algorithms that apply in probabilistic RL. This algorithm addresses the exploration-exploitation trade-off in decision-making processes [3]. This algorithm is based on a Bayesian approach, where the algorithm selects the arm that is

most probable to succeed. We use this algorithm with Laplace approximation to update the posterior. Those posterior distributions are complex with Gaussian distribution, and when sampling from them is difficult, we can use Laplace approximation to sample from them. So Thompson sampling with Laplace approximation enables us to treat uncertainty and sample from actions that are most likely to get a click [4], [5], [6].

2 Methodology

2.0.1 Contextual Linear Bandits

Contextual linear bandits are a powerful framework for recommender systems [7, 8]. In this context, the reward prediction is modeled as a linear function of the context vectors.

The reward R_t for time t is given by:

$$R_t = \mathbf{z}_t^T \boldsymbol{\theta}_{\mathbf{z}_t} + \epsilon_t \tag{1}$$

where:

- $\mathbf{z}_t \in \mathbb{R}^d$ is the context vector of the user at time t, with d being the dimensionality of the context.
- $\boldsymbol{\theta}_{\mathbf{z}_t} \in \mathbb{R}^d$ represents the context-specific parameter vector for the advertisement (arm) associated with the context $\mathbf{z}_t$. The dimensionality of $\boldsymbol{\theta}_{\mathbf{z}_t}$ is also d.
- ϵ_t is the noise term, assumed to be normally distributed with zero mean.

In our system, z_t is a known vector of features of the site that users arrive at, with the site arrivals following an exponential distribution. θ_{z_t} is an unknown latent vector of features of the ads (arms) that we need to learn. The expression $z_t^T \theta_{z_t}$ represents the click rate (reward r_t) of showing ads on the site.

2.1 Logistic Regression

In the context of modeling click-through rates (CTR), Bayesian methods provide a robust framework for incorporating prior beliefs and updating these beliefs with observed data.

2.1.1 Bayesian Methods

Bayesian statistical methods use Bayes' theorem to compute and update probabilities after obtaining new data. Bayes' theorem describes the conditional probability of an event based on data as well as prior information or beliefs about the event or conditions related to the event.[3][4] For example, in Bayesian inference, Bayes' theorem can be used to estimate the parameters of a probability distribution or statistical model. Since Bayesian statistics treats probability as a degree of belief, Bayes' theorem can directly assign a probability distribution that quantifies the belief to the parameter or set of parameters [9, 10].

Prior and Likelihood - **Prior**: The prior distribution encodes our beliefs about the parameters before observing any data. In this model, we use a multivariate Gaussian prior for the parameter vector $\boldsymbol{\theta}_z$. This Gaussian prior assumes that the parameters follow a normal distribution with a mean of zero and some covariance matrix, reflecting our initial uncertainty about the parameter values.

- **Likelihood**: The likelihood function describes the probability of the observed data given the parameters. Since the click-through rate is a probability that lies between 0 and 1, we use logistic regression to model this rate. Logistic regression maps the linear combination of the context vector $\mathbf{z}_t$ and the parameter vector $\boldsymbol{\theta}_z$ to a probability using the logistic function. Specifically, the likelihood of observing a click y_t given the context $\mathbf{z}_t$ and parameters $\boldsymbol{\theta}_z$ is given by:

$$P(y_t \mid \mathbf{z}_t, \boldsymbol{\theta}_z) = \text{Bernoulli}(\sigma(\mathbf{z}_t^T \boldsymbol{\theta}_z)) \tag{2}$$

where $\sigma(\cdot)$ denotes the logistic function:

$$\sigma(z) = \frac{1}{1 + e^{-z}} \tag{3}$$

Here, $\sigma(\mathbf{z}_t^T \boldsymbol{\theta}_z)$ maps the linear predictor $\mathbf{z}_t^T \boldsymbol{\theta}_z$ to the interval [0, 1], representing the probability of a click.

Laplace Approximation To approximate the posterior distribution of $\boldsymbol{\theta}_z$ given the data, we use Laplace approximation. This technique approximates the posterior distribution by a Gaussian centered at the mode of the posterior distribution. The steps involved are [11]:

1. **Compute the Mode**: Identify the mode of the posterior distribution, denoted as $\hat{\boldsymbol{\theta}}_z$. This is the parameter vector that maximizes the posterior probability. The mode can be found by solving:

$$\hat{\boldsymbol{\theta}}_z = \arg\max_{\boldsymbol{\theta}_z} P(\boldsymbol{\theta}_{\boldsymbol{z}} \mid \mathbf{z}_t, y_t) \tag{4}$$

   This is typically done by maximizing the log-posterior, which is:

$$\log P(\boldsymbol{\theta}_z \mid \mathbf{z}_t, y_t) = \log P(y_t \mid \mathbf{z}_t, \boldsymbol{\theta}_z) + \log P(\boldsymbol{\theta}_z) \tag{5}$$

   where $\log P(y_t \mid \mathbf{z}_t, \boldsymbol{\theta}_z)$ is the log-likelihood and $\log P(\boldsymbol{\theta}_z)$ is the log-prior.

2. **Approximate the Covariance**: Compute the covariance matrix of the approximate Gaussian distribution. This is obtained by evaluating the Hessian matrix of the negative log-posterior at $\hat{\boldsymbol{\theta}}_z$. The Hessian matrix $\mathbf{H}$ is given by:

$$\mathbf{H} = -\nabla^2 \log P(\boldsymbol{\theta} \mid \mathbf{z}_t, y_t)\Big|_{\boldsymbol{\theta}=\hat{\boldsymbol{\theta}}_z} \tag{6}$$

   where ∇^2 denotes the second derivative.

   The covariance matrix of the Gaussian approximation is then:

$$\mathbf{C} = \mathbf{H}^{-1} \tag{7}$$

3. **Gaussian Approximation**: The posterior distribution is approximated by a Gaussian distribution with mean $\hat{\boldsymbol{\theta}}_z$ and covariance $\mathbf{C}$. Thus, the approximate posterior distribution is:

$$P(\boldsymbol{\theta} \mid \mathbf{z}_t, y_t) \approx \mathcal{N}(\boldsymbol{\theta} \mid \hat{\boldsymbol{\theta}}_t, \mathbf{C}) \tag{8}$$

Laplace approximation simplifies inference by providing a closed-form Gaussian approximation to the complex posterior distribution, making it easier to perform Bayesian inference and model updates.

2.1.2 Modeling Click-Through Rates

In our system, logistic regression is applied to model the click-through rate (CTR) between 0 and 1. In Thompson sampling with Laplace approximation, we use a multivariate Gaussian prior and infer the posterior to learn the best estimate for the probability of a click-through rates.

2.1.3 Log-Likelihood Function

For logistic regression, the likelihood function for a single observation r_t given the feature vector z_t and parameter θ_z is:

$$p(r_t \mid \theta_z, z_t) = \sigma(z_t^\top \theta_z)^{r_t} \cdot (1 - \sigma(z_t^\top \theta_z))^{1-r_t}$$

where $\sigma(z_t^\top \theta_z) = \frac{1}{1+\exp(-z_t^\top \theta_z)}$ is the sigmoid function.

The log-likelihood for each observation is:

$$\log p(r_t \mid \theta_z, z_t) = r_t \log(\sigma(z_t^\top \theta_z)) + (1 - r_t) \log(1 - \sigma(z_t^\top \theta_z))$$

See [12] for more details on logistic regression.

2.1.4 Gradient of the Log-Likelihood

The gradient of the log-likelihood with respect to θ_z is:

$$\frac{\partial \log p(r_t \mid \theta_z, z_t)}{\partial \theta_z} = (r_t - \sigma(z_t^\top \theta_z)) z_t$$

2.1.5 Hessian of the Log-Likelihood

The Hessian matrix $H(\theta_z)$ is the second derivative of the log-likelihood with respect to θ_z:

$$H(\theta_z) = -\sigma(z_t^\top \theta_z)(1 - \sigma(z_t^\top \theta_z)) z_t z_t^\top$$

For the entire dataset with T observations, the Hessian for the log-posterior is:

$$H(\theta_z) = \sum_{t=1}^{T} \sigma(z_t^\top \theta_z)(1 - \sigma(z_t^\top \theta_z)) z_t z_t^\top + \Sigma_0^{-1}$$

where Σ_0 is the prior covariance matrix.

See [14] for a detailed discussion on Thompson Sampling.

2.1.6 Prior Distribution

Assuming a Gaussian prior for the parameter θ_z:

$$p(\theta_z) = \mathcal{N}(\theta_z \mid \mu_0, \Sigma_0)$$

Refer to [13] for details on Bayesian logistic regression.

2.1.7 Posterior Distribution

The posterior distribution is proportional to the product of the prior and likelihood:

$$p(\theta_z \mid \text{data}) \propto p(\theta_z) \prod_{t=1}^{T} p(r_t \mid \theta_z, z_t)$$

2.1.8 MAP Estimate

The MAP estimate of θ_z is found by maximizing the log-posterior:

$$\theta_z^{\text{MAP}} = \arg\max_{\theta_z} \left(\sum_{t=1}^{T} \left[r_t \log \sigma(z_t^\top \theta_z) + (1 - r_t) \log(1 - \sigma(z_t^\top \theta_z)) \right] - \frac{1}{2} (\theta_z - \mu_0)^\top \Sigma_0^{-1} (\theta_z - \mu_0) \right)$$

2.1.9 Hessian Matrix at MAP Estimate

The Hessian matrix at the MAP estimate θ_z^{MAP} is:

$$H(\theta_z) = -\sum_{t=1}^{T} \sigma(z_t^\top \theta_z)(1 - \sigma(z_t^\top \theta_z)) z_t z_t^\top - \Sigma_0^{-1}$$

2.1.10 Posterior Distribution (Laplace Approximation)

Using the Laplace approximation, the posterior distribution is approximated by a Gaussian distribution:

$$p(\theta_z \mid \text{data}) \approx \mathcal{N}(\theta_z^{\text{MAP}}, \Sigma_z)$$

where $\Sigma_z = H(\theta_z^{\text{MAP}})^{-1}$.

2.1.11 Updating Mean and Covariance

Each time new data (z_t, r_t) arrives:

2.1.12 Updating Mean and Covariance

Each time new data (z_t, r_t) arrives:

- **Update the MAP estimate** θ_z^{MAP}: Use gradient descent with a learning rate $\alpha = 0.01$ to iteratively maximize the log-posterior. The update rule for gradient descent is:

$$\theta_z^{(k+1)} = \theta_z^{(k)} + \alpha \cdot \frac{\partial \log p(r_t \mid \theta_z, z_t)}{\partial \theta_z}$$

  where $\theta_z^{(k)}$ represents the parameter values at iteration k.

- **Update the covariance matrix** Σ_z: Compute the Hessian at the new MAP estimate $\theta_z^{(k+1)}$ and take its inverse. The covariance matrix is:

$$\Sigma_z = \left[\sum_{t=1}^{T} \sigma(z_t^\top \theta_z^{(k+1)})(1 - \sigma(z_t^\top \theta_z^{(k+1)})) z_t z_t^\top + \Sigma_0^{-1} \right]^{-1}$$

2.2 Online Logistic Regression

Online logistic regression is an efficient method for updating model parameters in real-time as new data arrives. This approach is particularly useful in scenarios where data is continuously streaming, such as in recommender systems.

2.2.1 Model Description

In online logistic regression, the model is updated incrementally as each new data point (z_t, r_t) is observed. The core idea is to adaptively learn the parameters θ_z that maximize the likelihood of the observed data, given a logistic model. The logistic regression model estimates the probability of a binary outcome, such as a click-through rate in a recommender system, using the sigmoid function:

$$\sigma(z_t^\top \theta_z) = \frac{1}{1 + \exp(-z_t^\top \theta_z)}$$

where $\sigma(z_t^\top \theta_z)$ represents the probability of a positive outcome (e.g., click), and z_t is the feature vector at time t.

2.3 Online Logistic Regression

Online logistic regression is a method used for binary classification problems where the goal is to predict the probability of a binary outcome, such as whether a user will click on an ad. The logistic regression model is updated incrementally as new data arrives, making it well-suited for large-scale or streaming data scenarios.

2.3.1 Model Description

Given a context vector $\mathbf{z}_t \in \mathbb{R}^d$ at time t, we predict the probability p_t of a binary outcome $y_t \in \{0, 1\}$ using the logistic function [2]:

$$p_t = \sigma(\mathbf{z}_t^\top \boldsymbol{\theta}_t) = \frac{1}{1 + \exp(-\mathbf{z}_t^\top \boldsymbol{\theta}_t)}$$

where $\boldsymbol{\theta}_t \in \mathbb{R}^d$ is the parameter vector at time t that needs to be learned.

2.3.2 Loss Function

The loss function used in logistic regression is the negative log-likelihood, also known as the binary cross-entropy loss. For a single observation $(y_t, \mathbf{z}_t)$, the loss function is given by:

$$\ell(\boldsymbol{\theta}_t) = -\left[y_t \log(\sigma(\mathbf{z}_t^\top \boldsymbol{\theta}_t)) + (1 - y_t) \log(1 - \sigma(\mathbf{z}_t^\top \boldsymbol{\theta}_t))\right]$$

Expanding this, the loss function can be expressed as:

$$\ell(\boldsymbol{\theta}_t) = -y_t \log\left(\frac{1}{1 + \exp(-\mathbf{z}_t^\top \boldsymbol{\theta}_t)}\right) - (1 - y_t) \log\left(1 - \frac{1}{1 + \exp(-\mathbf{z}_t^\top \boldsymbol{\theta}_t)}\right)$$

This loss function measures the discrepancy between the predicted probability $\sigma(\mathbf{z}_t^\top \boldsymbol{\theta}_t)$ and the true label y_t.

2.3.3 Gradient of the Loss Function

The gradient of the loss function with respect to the parameter vector $\boldsymbol{\theta}_t$ is used to update the parameters in gradient descent. The gradient for the loss function at time t is:

$$\nabla_{\boldsymbol{\theta}_t} \ell(\boldsymbol{\theta}_t) = (\sigma(\mathbf{z}_t^\top \boldsymbol{\theta}_t) - y_t)\mathbf{z}_t$$

Here, $\sigma(\mathbf{z}_t^\top \boldsymbol{\theta}_t)$ is the predicted probability, and y_t is the actual label. The difference $\sigma(\mathbf{z}_t^\top \boldsymbol{\theta}_t) - y_t$ represents the prediction error.

2.3.4 Gradient Descent for Optimization

In the online setting, the parameters $\boldsymbol{\theta}_t$ are updated iteratively as new data points arrive. The update rule using stochastic gradient descent (SGD) is given by:

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \eta \nabla_{\boldsymbol{\theta}_t} \ell(\boldsymbol{\theta}_t)$$

where $\eta > 0$ is the learning rate, which controls the step size of the update. Substituting the gradient, the update rule becomes:

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \eta(\sigma(\mathbf{z}_t^\top \boldsymbol{\theta}_t) - y_t)\mathbf{z}_t$$

This update rule ensures that the parameter vector $\boldsymbol{\theta}_t$ is adjusted in the direction that reduces the prediction error, thereby improving the model's accuracy over time.

2.3.5 Advantages of Online Logistic Regression

Online logistic regression provides several advantages for real-time applications:

- **Scalability**: It efficiently handles large-scale datasets by processing one data point at a time.
- **Adaptability**: The model can quickly adapt to new data, which is crucial in dynamic environments like recommender systems.
- **Low Memory Usage**: Only a small subset of the data is kept in memory at any given time, reducing the overall memory footprint.

2.3.6 Implementation Details

The implementation of online logistic regression involves the following steps:

- Initialize the parameter vector θ_z to zero or a small random value.
- For each new data point (z_t, r_t):

- Compute the predicted probability using the sigmoid function.
- Update the parameter vector θ_z using the stochastic gradient descent rule.

Online logistic regression is particularly effective in applications where the data arrives sequentially and where the model needs to be continuously updated. For more information on online learning algorithms and logistic regression, refer to [**?**] and [12].

3 Experiments and Results

4 Results

In this problem, we focus on optimizing ad displays using a known feature vector $\mathbf{z}$ with dimension d, where each element of $\mathbf{z}$ follows a binomial distribution with values 0 or 1. For the ads, the parameter $\boldsymbol{\theta}_z$ is unknown, and we aim to learn its distribution. The distribution of $\boldsymbol{\theta}_z$ is modeled as a multivariate normal distribution with mean $\boldsymbol{\mu}$ (a d-dimensional vector) and an identity covariance matrix $\mathbf{I}_d$.

Two algorithms were used to update the posterior distribution of the parameters, with batch sizes of 100 and 1000, respectively. The learning rate for updating the parameter α was set to 0.01.

4.1 Mathematical Formulation

- **Known Feature Vector**: $\mathbf{z} \in \mathbb{R}^d$, where $z_i \sim \text{Binomial}(1, p)$
- **Unknown Parameter for Ads**: $\boldsymbol{\theta}_z \in \mathbb{R}^d$, where $\boldsymbol{\theta}_z \sim \mathcal{N}(\boldsymbol{\mu}, \mathbf{I}_d)$
- **Mean Vector**: $\boldsymbol{\mu} \in \mathbb{R}^d$
- **Covariance Matrix**: $\mathbf{I}_d$ (Identity matrix of size $d \times d$)

The mean vector $\boldsymbol{\mu}$ is updated using the following equation:

$$\boldsymbol{\mu}_{\text{new}} = \boldsymbol{\mu}_{\text{old}} + \alpha \cdot \nabla_{\boldsymbol{\mu}} \mathcal{L}$$

where $\alpha = 0.01$ is the learning rate, and $\nabla_{\boldsymbol{\mu}} \mathcal{L}$ is the gradient of the loss function $\mathcal{L}$ with respect to $\boldsymbol{\mu}$.

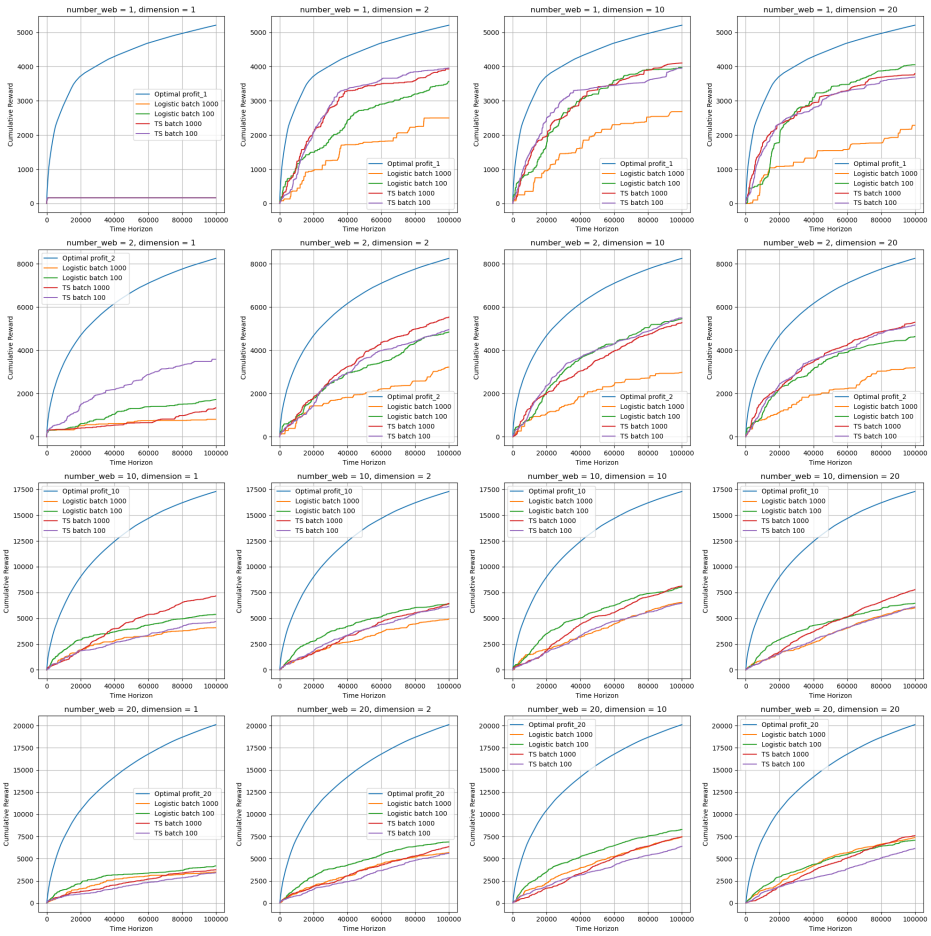

Figure 1: Visualization of the scenario with 200 ads. The rows represent the number of websites, and the columns correspond to the dimensions of the latent features.

We observed that in logistic regression, using a batch update size of 100 working better results compared to a batch size of 1000. Specifically, the smaller batch size allowed for more frequent updates, leading to a more responsive model that could adapt quickly to changes, thereby increasing profit more effectively than the larger batch size. However, when using Thompson sampling, the probabilistic nature of the model showed that a batch size of 1000 was more effective in maximizing profit.

This suggests that probabilistic models like Thompson sampling are more

robust in this scenario, where the ability to explore different strategies and adapt quickly is crucial. The model's inherent randomness allows it to better navigate the exploration-exploitation trade-off, leading to improved performance in terms of profit maximization compared to logistic regression, especially with the smaller batch size.

5 Conclusion

In our analysis, we observed that increasing the dimension of latent features generally improves the maximum profit and enhances the learning distribution. However, this trend does not continue indefinitely. Beyond a certain dimension, the profit or accuracy of learning may no longer increase and can even deteriorate.

This suggests that while higher-dimensional features can provide more information and improve model performance, there is a trade-off. After reaching an optimal dimension, further increases in dimensionality may lead to overfitting or increased complexity without a corresponding improvement in performance.

Therefore, when determining the best dimension for the latent features, it is crucial to balance the benefits of higher dimensionality with the potential risks of deteriorating algorithm performance. Careful consideration and validation are needed to ensure that the dimensionality chosen does not adversely affect the effectiveness of the algorithm. Key Points in the Conclusion:

6 References

References

[1] L. Bottou, *Online Learning and Stochastic Approximations*, Cambridge University Press, 1998.

[2] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*, Springer, 2009.

[3] S. Agrawal, and N. Goyal, "Analysis of Thompson Sampling for the Multi-Armed Bandit Problem," *Journal of Machine Learning Research*, vol. 23, 2012, pp. 39.

[4] D. J. C. MacKay, "Bayesian Interpolation," *Neural Computation*, vol. 4, no. 3, 1992, pp. 415-447.

[5] C. E. Rasmussen, and C. K. I. Williams, *Gaussian Processes for Machine Learning*, MIT Press, 2006.

[6] D. J. Russo, B. Van Roy, A. Kazerouni, I. Osband, and Z. Wen, "A Tutorial on Thompson Sampling," *Foundations and Trends® in Machine Learning*, vol. 11, no. 1, 2018, pp. 1-96.

[7] M. L. Puterman, *Markov Decision Processes: Discrete Stochastic Dynamic Programming*, Wiley, 1994.

[8] R. Burke, *Hybrid Recommender Systems: Survey and Experiments*, User Modeling and User-Adapted Interaction, vol. 12, no. 4, pp. 331-370, 2002.

[9] R. Burke, *Gelman, Andrew; Carlin, John B.; Stern, Hal S.*; Dunson, David B.; Vehtari, Aki; Rubin, Donald B. (2013). Bayesian Data Analysis (Third ed.). Chapman and Hall/CRC. ISBN 978-1-4398-4095-5.

[10] R. Burke, *McElreath, Richard (2020). Statistical Rethinking* : A Bayesian Course with Examples in R and Stan (2nd ed.). Chapman and Hall/CRC. ISBN 978-0-367-13991-9..

[11] R. Burke, *Kass, Robert E.; Tierney, Luke; Kadane, Joseph B. (1991).* : Laplace's method in Bayesian analysis". Statistical Multiple Integration. Contemporary Mathematics. Vol. 115. pp. 89–100. doi:10.1090/conm/115/07. ISBN 0-8218-5122-5.

[12] D. C. Montgomery, E. A. Peck, and G. G. Vining, *Introduction to Linear Regression Analysis*, Wiley, 2021.

[13] C. M. Bishop, *Pattern Recognition and Machine Learning*, Springer, 2006.

[14] *A Tutorial on Thompson Sampling Daniel Russo, Benjamin Van Roy, Abbas Kazerouni, Ian Osband, Zheng Wen.*