# Optimization of the Multi-armed Bandit Problem with Graphical Models: a Bayesian Perspective

March ２０１９

Zhao Chen

# Optimization of the Multi-armed Bandit Problem with Graphical Models: a Bayesian Perspective

Graduate School of Systems and Information Engineering

University of Tsukuba

March ２０１９

Zhao Chen

**Abstract**

This dissertation handles difficulty of accurately modeling stochastic data patterns based on discrete data samples. In particular, we investigate the multi-armed bandit (MAB) problem which consists of a collection of random variables called *arms*, each with unknown probability distribution. The core task is to design optimal strategy of quickly exploring the best arm, i.e. the one of the maximum expected value, meanwhile maintaining a minimized loss, or *regret*, commonly computed as the difference between the sum of sampled rewards and the sum of rewards under ideal strategy. Simple as the problem might seem, numerous efforts from research community have been devoted to improving its learning strategies due to its generalizability to a wide range of problems involving resource allocation such as clinical trials, adaptive routing, online ads, etc. Mostly notably, it leads to crucial implication for how stepwise decision making can be optimized in the sense that the MAB setting is essentially a one-state Markov decision process, a simplified scenario that in practice many more complicated reinforcement learning problems get broken down to. In this thesis, we start with recap of commonly used classical approaches of solving the MAB problem and then investigate some details of Gaussian regression based methods from previous study which achieves decent asymptotical regret bounds. Next to literature study, we propose our new solution to the MAB problem under stochastic environment setting. Concretely, we design a novel graphical model from the perspective of Bayesian machine learning where samples collected from arms are used as training data and the set of choosable arms are treated as action space the model is expected to predict within. Our solution also provides a new sampling technique based on *Thompson sampling*, a classical heuristic that autonomously balances the exploration-exploitation trade-off in decision-making problems with uncertainty. Experimental study reveals that given graphical model inference and proper hyperparameter tuning, the Thompson sampling variation delivers more satisfactory overall prediction compared to conventional Bayesian posterior sampling methods. The main contribution of this work is a system of problem independent learning techniques applicable to all scenarios where solution to the MAB problem is desired.

# Acknowledgements

First and foremost I would like to thank my supervisor Professor Takehito Utsuro for his academic support and engagement in my degree program during the recent three years, without whose supervision I would not have smoothly accomplished the degree requirement as I have.

I wish to express special gratitude to Dr. Bin Yang and Mr. Kohei Watanabe, two of my coworkers who provided solid guidance and theoretical discussions on this work during my internship position at Rakuten Institute of Technology. Without their effort, this research would have been out of the question.

Furthermore I feel like explicitly mentioning one of my close friends, Miao Jiang who comes from technology industry, for my sincere appreciation on his career and academic advice along with persistent encouragement on my career path throughout these years. Without his help it would have been far more difficult for me to constantly stay tuned with the frontiers of both the industry and academia.

Last but not least, words are not sufficient to cover my thankfulness to everyone who came to my aid, shed light on my inspiration of this research or offered generous opportunity for me to make progress on this work including my families, friends, faculty of Department of Intelligent Interaction Technologies, Systems and Information Engineering, University of Tsukuba and staff in Rakuten Institute of Technology.

# Contents


**1 Introduction** **6**

**2 Background and Theoretics** **10**

2.1 Two-armed Bandit and A/B Testing . . . . . 10

2.2 The Multi-armed Bandit Problem . . . . . 14

2.2.1 Problem Definition . . . . . 14

2.2.2 Useful Theorems . . . . . 16

2.2.3 Classical Approaches – Bernoulli Rewards . . . . . 17

2.2.4 Classical Approaches – Gaussian Rewards . . . . . 27

2.3 Gaussian Process Regression . . . . . 29

2.3.1 Introduction to Gaussian Process Regression . . . . . 29

2.3.2 GP-UCB Algorithm – Regression Under Bandit Setting . . . . . 31

2.3.3 GP-UCB – Information Gain and Regret Bounds . . . 32

2.4 Probabilistic Graphical Models . . . . . 38

2.4.1 Directed Models - Bayesian Network . . . . . 39

2.4.2 Undirected Models - Markov random field . . . . . 45

2.4.3 Inference Algorithms . . . . . 49

**3 Optimizing Bandit Problems with Graphical Models** **55**

3.1 Introduction on Contribution . . . . . 56

3.2 Model Declaration . . . . . 58

3.2.1 Problem Nature . . . . . 58

3.2.2 Graphical Representation . . . . . 58

**4 Graphical Inference** **64**
4.1 Parameter Estimation . . . . . 64
4.2 Relevance to the Bandit Problem . . . . . 71

**5 Graphical Learning** **72**
5.1 Decision Making Policy . . . . . 73
5.1.1 Acquisition Function . . . . . 74
5.1.2 Epsilon Greedy . . . . . 75
5.1.3 Thompson Sampling . . . . . 75

**6 Experiments and Evaluation** **77**
6.1 Experiment Setup . . . . . 78
6.1.1 Hyperparameter Tuning . . . . . 78
6.1.2 Experiment 1 - Learning Synthetic Functions . . . . . 79
6.1.3 Experiment 2 - Learning Recommendation Scheduling . . . . . 81
6.2 Evaluation . . . . . 84
6.2.1 Synthetic Data Evaluation . . . . . 84
6.2.2 Testbench Data Evaluation . . . . . 86
6.3 Real-life Dataset . . . . . 88
6.4 Further Discussion . . . . . 91
6.4.1 Sampling Complexity . . . . . 91
6.4.2 Numerical Stability . . . . . 92

**7 Concluding Chapter** **94**
7.1 The Bandit Problem and Markov Decision Process . . . . . 95
7.2 The Bandit Problem and Reinforcement Learning . . . . . 96
7.3 Summary . . . . . 97

# List of Figures


2.1 Bayesian network example . . . . . 39
2.2 Generative Model vs. Discriminative Model . . . . . 44
2.3 Markov network example . . . . . 46

3.1 Markov property . . . . . 59
3.2 Bayesian property . . . . . 60
3.3 Final Graphical Model - Gaussian Markov Random Field . . . 62

6.1 1D Test - Synthesized Environment . . . . . 85
6.2 2D Test - Synthesized Environment . . . . . 85
6.3 3D Test - Synthesized Environment . . . . . 86
6.4 1D Test - Test Bench Environment . . . . . 87
6.5 2D Test - Test Bench Environment . . . . . 87
6.6 3D Test - Test Bench Environment . . . . . 88
6.7 Ground Truth Environment for the Dataset . . . . . 89
6.8 Click Loss by Days . . . . . 90

# List of Tables

7.1 Pros and Cons between GP and GMRF . . . . . 95
7.2 Bandit and Reinforcement Learning Problems . . . . . 97

# Chapter 1

# Introduction

The stochastic multi-armed bandit (MAB) problem is a sequential decision-making problem and essential for solving subparts of many more complex stochastic decision problems in practice. The problem has a simple statement that given a collections of random variables with absolutely no prior knowledge about their probability distribution, solution is desired to find the best of them, i.e., the one with maximum expected value. Intuitively, a primary option of solving this problem resorts to sampling-based statistical modeling which persists in collecting data from the given random variables until either enough knowledge is acquired to reconstruct their true distributions (in case of parametric inference) or at least statistically significant differences are discovered among random variables (in case of hypothesis testing). In whichever case we step towards gaining confidence to evidently account for why some random variable has a larger mean above the others. Unfortunately, sampling from practical data sources faces complex constraints which frequently prohibit large scale data collection process in real life scenarios. Consider a clinical trial involving a list of experimental treatments with the objective of figuring out the most effective treatment method, i.e., treatment with the highest rate of success. Underoptimized tests in this scenario easily lead to unwanted victims who would suffer from inappropriate treatment which in extreme cases indicates loss of human lives and such fatal failure could have been avoided with proper strategy of conducting experiments. Apparently, for this particular problem the number of patients who get improper treatment shall in the highest priority be minimized while the most reliable treatment is still to be accurately determined. The MAB setting typ-

ically applies to this problem in such a way that candidate therapies/drugs constitute the collection of unknown random variables whose probability distributions are yet to be inferred; in this case the probability refers to success rate of achieving medical effectiveness. The critical resource is without doubt the patients involved in this experiment. In general a MAB problem is all about making optimization concerning the cost of critical resource and accuracy of probability inference. An ideally optimal strategy is expected to quickly find out the best random variable at minimum cost required to achieve that. In the clinical trial example, we intend to come to correct conclusion on which type of candidate treatment is the best with the fewest possible subject patients involved in this experiment. Similar optimization problems with resource constraints are also seen in various other applications such as adaptive routing where a router dynamically forwards data packets through several different candidate routes trying to find the route that maximizes network throughput. Another much more viable example is online advertisement recommendation under fixed marketing budget that tries to decide which one out of all the candidate ads attracts the most attention from customers.

Statistical hypothesis testing is a category of commonly used classical approaches of comparing the "quality" of unknown random variables. By evenly collecting data referred to as "evidence" from random variables, it computes some statistics called $P$-value based on evidence and compares $P$-value to some predefined significance level to infer which variable is better. Statistical hypothesis testing fits well in applications without hard constraints on laxity of conclusion. For instance, in order to conduct the hypothesis test whether an ad post attracts more visitors with its refreshed design, statistics of user responses to both the old and new versions need to be collected within some frame of time intervals. A noticeable problem is that such an experiment makes no valid conclusion until it finishes collecting all the necessary data to decide whether or not the hypothesis is to be rejected. In many use cases we hope to extract partial information during data collection process instead of analyzing everything afterwards, i.e., an online approach is requested so that we are able to perform incremental update to our knowledge as data collection is going on. The major benefit of online approaches is that immediate action can be taken to reduce the loss caused by suboptimal decisions during data collection process. In the example of redesigned ad, if traffic gets increasingly redirected to the more popular version of ad as soon

as we acquire some amount of data, overall losses of potential visitors can be reduced compared to evenly distributing traffic to both versions throughout the experiment. A fundamental approach of making serial updates to statistical metrics on the fly is Bayesian paradigm which replaces the conceptually frequentist $P$-value with likelihood computed from currently available data and updates the posteriors on the fly as new data gets collected. While Bayesian analysis does satisfy the online characteristic it still fails to answer one of the most difficult questions in stochastic comparison - the exploration-exploitation dilemma. The problem basically focuses on strategy of how data gets sampled during Bayesian analysis. As for the ad example again, given fixed budget of displaying ads, the optimal strategy of allocating traffic to different ads can be hard to figure out. Naive Bayesian analysis tends to greedily trust current posteriors and this easily leads to incorrect inference if data sampled at early stage tends to favor the less popular candidate ad. It is a typical pure "exploitation" case in which we choose to trust whatever knowledge we have at current moment with no interest in discovering better alternatives. Oppositely, in a typical pure "exploration" case, uniformly random sampling is performed so that both ads share the same probability of getting displayed. Pure exploration completely ignores knowledge from data and in general produces unoptimized overall loss. There exists abundant literature studying the trade-off between exploitation and exploration in Bayesian data analysis with similar purpose of making optimal decisions at minimal cost. This thesis takes comprehensive study on the MAB problem as a game of "guessing" the best random variable, as a high-level abstraction of lots of real world decision-making problems.

The thesis consists of three primary partitions. The first partition is responsible for systematic background introduction on various aspects of theoretical foundation based on which this research is derived. In addition to preliminary introduction to the problem definition starting with a simplified two-armed scenario, a few typical classical methods are introduced. It also comes with discussion on asymptotic regret characteristics under different methods and illustration on how regret bounds are proved. In particular it also reviews how Gaussian process was used for achieving novel convergence rates of regret in previous works and in which way this research is inspired from them. Aside from the bandit problem itself, the background introduction incorporates considerable coverage on basics of probabilistic graphical models, a category of graph-based predictive models with wide deployment

in machine learning applications; how graphical models are defined to interpret stochastic processes and get connected to real-world problems; and brief description on common graphical inference algorithms. The second primary partition of this thesis is definition of the proposed model in full detail as the core part of the research. Our contribution includes a general inference model of predicting reward patterns in the MAB problem assuming a collection of possibly correlated arms. The proposed solution considers the candidate arms as an output sequence from a "black-box" function pending for global optimization as opposed to isolated subjects in hypothesis testing, so that interpolation can be applied to learn such a function through samples as training data. The third partition covers experimental design for evaluating different strategies of drawing data samples from the black-box function and provides some numerical guidelines on analyzing posterior distribution of rewards in order to efficiently achieve balance between exploration and exploration.

In the concluding chapter, insight of the MAB problem towards general decision-making process and reinforcement learning is briefly discussed. We believe that this work brings an alternative perspective of a classic problem that has no perfect solution, whose intuition is so simple that not only computers but also humans are facing it everyday whenever it comes to uncertainty. According to Christian [1] – What should we do, and leave undone, in a day or in a decade? What degree of mess should we embrace - and how much order is excessive? What balance between *new* experience and *favored* ones makes for the most fulfilling life?

# Chapter 2

# Background and Theoretics

All background study on existing works and theoretics of interest for this research are organized in this chapter. The MAB problem came to existence out of many occasions where hypothesis tests such as A/B tests are poorly applicable. Formal definition of the problem nature is given along with typical classical solutions. We put emphasis on regret bound analysis to illustrate that how strategy on exploration-exploitation trade-off brings significant impact on asymptotic behavior of regret and why even minor optimization makes a difference. Then we make at-length introduction on some previous state-of-the-art approach called GP-UCB which is based on Gaussian process regression, and its improved upper bound of regret. Next to the MAB problem, an introductory section is prepared for quick review on probabilistic graphical models, their declarative representation and graphical inference algorithms.

## 2.1 Two-armed Bandit and A/B Testing

A particular case of the MAB problem is the two-armed bandit, which states that a slot machine has two arms for gamblers to play with and pulling each arm delivers random rewards of distinct pattern. The simplest analogy is a coin tossing game. Suppose two biased coins are present and we are interested in knowing which of the two coins tends to be further biased towards heads standing for reward 1 as opposed to tails for reward 0. Assuming coin 1 gets tossed for $n_1$ trials and coin 2 for $n_2$ trials, we know that the

total rewards from both coins are in binomial distribution of $B(n_1, p1)$ and $B(n_2, p2)$, where $p_1$ and $p_2$ are unknown probabilities of heads. In an attempt to test out whether $p_1$ and $p_2$ are significantly different from each other, the $t$-test can effectively be applied because long term $p_1$ and $p_2$ are asymptotically Gaussian [2], an assumption held by the $t$-test. The following null hypothesis is proposed.

$$H_0 : p_1 = p_2$$

The relevant alternative hypothesis is listed as rival to $H_0$.

$$H_1 : p_1 < p_2 \; or \; p_1 > p_2$$

The $t$-test calculates what is called $t$-statistic as $T$ for metrics of deciding whether null hypothesis is significantly true or false;

$$T = \frac{E[X_1] - E[X_2]}{\sqrt{\frac{s_p^2}{n_1} + \frac{s_p^2}{n_2}}}$$

where $E[X_1]$ and $E[X_2]$ are empirical values of head probability that are in practice computed as ratio of observed heads to total tosses on a coin. Additionally pooled variance $s_p$ is requested for two-sample $t$-tests and is easily accessible as below.

$$s_p = \sqrt{\frac{\sum_{i=1}^{n_1}(X_1^{(i)} - E[X_1])^2 + \sum_{i=1}^{n_2}(X_2^{(i)} - E[X_2])^2}{n_1 + n_2 - 2}}$$

where $X_1^{(i)} = 1$ denotes a head at the $i$th trial on coin 1 and $X_1^{(i)} = 0$ otherwise. We reject the null hypothesis $H_0$ if $|T| > t_{n_1+n_2-2}(\frac{\alpha}{2})$ where $\alpha$ is a preselected significance level with common values of 0.05 or 0.01 and $t_{n_1+n_2-2}(\frac{\alpha}{2})$ is a threshold beyond which one-tailed cumulative density of $t$-distribution exceeds $\frac{\alpha}{2}$. As $t$-distribution is symmetric and resembles normal distribution except for that it has only one parameter called *degree of freedom* given as constant $n_1 + n_2 - 2$ in our example, $|T| > t_{n_1+n_2-2}(\frac{\alpha}{2})$ indicates that the empirical $t$-statistic falls within either of the two tails the sum of whose cumulative density is below the significance level $\alpha$. Hence it is highly unlikely that $p_1$ and $p_2$ are the same. Given that $H_0$ is rejected, we further accept the alternative hypothesis that $p_1 < p_2$ if $T < -t_{n_1+n_2-2}(\frac{\alpha}{2})$ or $p_1 > p_2$

if $T > t_{n_1+n_2-2}(\frac{\alpha}{2})$. Conclusion [1] on hypothesis testing does not reveal the absolute truth of candidate random variables but only tells whether we are confident enough to reject some proposed hypothesis at confidence level of $1-\alpha$ with the data we collected so far.

It is clear that the testing process described above does not come to any conclusion until it finishes all the trials on both coins. If we intend to accumulate knowledge as the coin tossing experiment goes on or for any practical reason data sizes $n_1$ and $n_2$ cannot be determined in advance, sequential analysis is needed. Bayesian estimation is a quick and simple fix to the problem of sequential A/B testing. Let a biased coin have head probability $p$ and $\boldsymbol{x} = \{x_1, x_2, ...x_i, ...\}$ be the list of trials where $x_i = 1$ for head and $x_i = 0$ for tail. Our task is to figure out the most probable $p$ based on observations we are able to collect. Following Bayesian paradigm we set up the likelihood function $\mathcal{L}(\boldsymbol{x}, p)$ given events and parameter $p$.

$$\mathcal{L}(\boldsymbol{x}, p) = Pr(\boldsymbol{x})Pr(p|\boldsymbol{x}) = Pr(p)Pr(\boldsymbol{x}|p)$$

Since a single of toss a coin is a Bernoulli trial, the event likelihood $Pr(\boldsymbol{x}|p)$ is product of independent Bernoulli probability. The marginal likelihood $Pr(\boldsymbol{x})$ is independent of parametric hypothesis so it is the same for any observation and will be treated as normalizing constant.

$$Pr(p|\boldsymbol{x}) = \frac{Pr(p)Pr(\boldsymbol{x}|p)}{Pr(\boldsymbol{x})} \propto Pr(p)\prod_{i=1}^{N} p^{x_i}(1-p)^{1-x_i}$$

The conjugate prior of Bernoulli probability is Beta distribution of density function [2] $f(x)$.

$$f(x) = \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)+\Gamma(\beta)}x^{\alpha-1}(1-x)^{\beta-1}, 0 < x < 1$$

Now replace the parameter prior $Pr(p)$ with Beta density ignoring constant parts.

$$Pr(p|\boldsymbol{x}) \propto p^{\alpha-1}(1-p)^{\beta-1}\prod_{i=1}^{N} p^{x_i}(1-p)^{1-x_i}$$

[1] In a majority of cases the terminology "$P$-value" refers to the probability of obtaining a test statistic that is equal to or more extreme than actual observation, which in our case equals $2\int_T^{+\infty} f_{n_1+n_2+1}(x)dx$, $f_{n_1+n_2+1}(x)$ being probability density function of $t$-distribution. But many literatures refer to $P$-value with alternative definitions making its precise meaning controversial so this thesis avoids usage of this terminology.

$$\propto p^{\alpha+\sum_{i=1}^{N} x_i-1}(1-p)^{\beta-1+N-\sum_{i=1}^{N} x_i}$$

It turns out that given observations $\boldsymbol{x}$, the posterior distribution of parameter $p$ is in the form of Beta distribution. Therefore the maximum likelihood estimation of posterior is the Beta mean as predictive $\hat{p}$.

$$\hat{p} = \frac{\alpha + \sum_{i=1}^{N} x_i}{\alpha + \sum_{i=1}^{N} x_i + \beta + N - \sum_{i=1}^{N} x_i} = \frac{\alpha + \sum_{i=1}^{N} x_i}{\alpha + \beta + N} \tag{2.1}$$

In practice Beta parameters $\alpha$ and $\beta$ are both initialized with 1. Equation 2.1 shows that only the total count of heads and number of trials $N$ at test time is required to estimate head probability of a biased coin. It is reasonable to see that $\hat{p} = 0.5$ before any test starts as $N = 0$ since by default we assume a coin has equal head/tail probability without observation. In the two-coin scenario we directly compare posterior predictive to decide which coin delivers higher rewards and such comparison is continuously available during the process of drawing samples because $\hat{p}$ is sequentially updated. However problems still exist for practical implementation with Bayesian A/B testing.

The above experiment setup fails to answer several crucial questions - when do we stop our experiment? While $N$ does not have to be defined in advance we need explicit criteria as stop signals. Early stopping may lead to overconfidence in biased samples. Over-experimenting leads to unnecessarily redundant waiting; extra cost can be another problem if playing the "game" of tossing coins is not free, which is unfortunately true in a lot of real-life cases. Moreover, it is an interesting question whether there exist better options than uniformly sampling among multiple random variables. To be specific, in the two-coin test we believe that it is helpful to make decisions on which coin to toss based on posterior $\hat{p}$ instead of taking turns tossing one and the other for subsequent trials. These questions bring forward typical thoughts on the aforementioned exploitation-exploration trade-off. In particular we are only willing to play minimized trials of tosses that are just enough to determine the head probability for every coin.

Coin tossing games are a simplified analogy to a lot of real-world applications that require Bayesian optimization such as regression analysis and sometimes parameter tuning for machine learning models. Perhaps the most commonly discussed application is online advertising where a user decides whether or not to click an ad. A one-time user response is a Bernoulli trial so that the number of user clicks an ad receives forms binomial distributions.

Typically more than two "coins" are being tossed in online advertising so we need much more sophisticated algorithm to predict how many clicks each ad is going to attract. Since marketing cost is charged for deploying ads, we also demand that the algorithm allow for additional budget constraint. Making such optimization brings forward the MAB problem with budget limits, the core problem this thesis focuses on solving.

## 2.2 The Multi-armed Bandit Problem

This section defines the MAB problem, discusses some of its classical algorithms and makes analytical discussion on how and why policy optimization affects regret bounds. It focuses on two types of reward pattern: Bernoulli and normally distributed rewards.

### 2.2.1 Problem Definition

Formal definition of the stochastic multi-armed bandit problem can be formulated as a sequential optimization problem consisting of $K$ random variables $\langle X_1, X_2, ..., X_K \rangle$ with associated means $\langle \mu_1, \mu_2, ..., \mu_K \rangle$ and variances $\langle \sigma_1^2, \sigma_2^2, ..., \sigma_K^2 \rangle$. For historical reasons random variables are called *arms* as the value of $X_i$ is conventionally interpreted as the reward from a slot machine when its $i$th arm is pulled. A player is allowed to make gambling decisions through a sequence of turns and may choose any of the given arms with free will at turn $t = 1, 2, 3, ....$ The player is motivated to find out the arm of the highest mean through sequential attempts while collecting a maximized amount of reward during the process. Bandit algorithms are used to design optimal strategies for such a goal.

The most common metric of evaluating bandit algorithms is expected cumulative regret. For any total $T$ turns, the expected cumulative regret under policy $\pi$ is defined as $R_T$

$$R_T = T\mu^* - \sum_{t=1}^{T} \mu_{\pi(t)}$$

where $\mu^* = max_i \mu_i$ and $\pi(t)$ is the index of choice at time $t$ subject to policy $\pi$. The expected cumulative regret basically tells us how much loss

is caused by policy $\pi$ after $T$ turns of bandit trials due to failure in finding the optimal strategy. An optimal strategy is an ideal policy that is aware of the true distribution of arm rewards so that the best arm is guaranteed to get selected at every turn. Thereby the cumulated expected reward from an optimal strategy is $T\mu^*$. In contrast realistic strategy has expected reward $\sum_{t=1}^{T} \mu_{\pi(t)}$ and in suboptimal cases $\mu_{\pi(t)} < \mu^*$. An alternative form [3] of $R_T$ can be expressed as:

$$R_T = \mathbb{E}\big[\sum_{t=1}^{T} \mu^* - \mu_{\pi(t)}\big] = \sum_{k=1}^{K} (\mu^* - \mu_{\pi(t)}) \mathbb{E}\big[N_k(T)\big]$$

where $N_k(T)$ denotes the number of trials on arm $k$ during the total $T$ turns. A classical conclusion by Lai & Robbins [4] states that under any uniformly good policy[2], $N_k(T)$ satisfies the following asymptotic lower bound.

$$\lim_{T\to\infty} \frac{\mathbb{E}[N_k(T)]}{log(T)} \geq \frac{1}{D_{KL}(x_k, x_*)} \ (\mu_k \neq \mu^*) \tag{2.2}$$

is true for $k$ such that $\mu_k < \mu^*$, where $D_{KL}$ denotes Kullback-Leibler divergence.

$$D_{KL}(x_k, x_*) = \int x_k \ln \frac{x_k}{x_*}$$

Since $D_{KL}(x_k, x_*)$ measures the difference from the distribution of a suboptimal arm $k$ to the distribution of the optimal arm, the lower bound of $N_k(T)$ implies that a good policy with no assumption on parameters of arms tends to better favor suboptimal arms with distributions closer to the optimal arm than those farther away. An intuitive explanation is that the closer a suboptimal distribution $x_k$ is to $x_*$, the more data is required to gain confidence on verifying that $x_*$ is indeed better than $x_k$.

The conclusion in 2.2 thus bounds the expected cumulative regret $R_T$ as well. Simply put, a uniformly good strategy in general MAB setting has its regret growth at least logarithmically, or $\mathbb{E}[R_T] \sim \Omega(log(T))$. A policy $\pi$ is said to efficiently solve the MAB problem if it achieves such a logarithmic lower bound, or $\mathbb{E}[R_T] \sim O(log(T))$.

The MAB problems in general do not make assumption on distribution categories of candidate arms $\langle X_1, X_2, ..., X_K \rangle$. Drawing reward samples repeatedly from the same arm $X_i$ generates an i.i.d sequence distributed under

[2]Policy $\pi$ is said to be "uniformly good" if $\mathbb{E}[N_k(T)] = o(T^n)$ for $n > 0$ and $\mu_k \neq \mu^*$ holds independently from problem setting.

mysterious distribution. However discussion in a majority of literature is focused on Bernoulli rewards restricting every $X_i$ in support $\{0, 1\}$. But this thesis puts emphasis on rewards in normal distribution with an effort of groping a wider range of application with bandit algorithms. The remaining parts of this section discusses typical classical approaches of solving the MAB problem of both Bernoulli and normally distributed rewards.

### 2.2.2 Useful Theorems

A few propositions are listed in this subsection as auxiliary for analyzing regret bounds of bandit algorithms to be discussed in subsequent parts of this section. The listed propositions will be directly used as theorems without proof in this thesis.

**Theorem 2.2.1.** *(Chernoff-Hoeffding Bound) Let $X_1, X_2, ..., X_K$ be independent random variables with support in $[0, 1]$ and $\mu = \mathbb{E}\big[\frac{1}{K}\sum_{i=1}^{K} X_i\big]$. Let $S_n$ be the empirical sum of $X_i$ so $S_n = \sum_{i=1}^{K} X_i$. Then for $a > 0$*

$$\begin{aligned} Pr\big\{S_n \geq K\mu + a\big\} &\leq \exp^{-2a^2/n} \\ Pr\big\{S_n \leq K\mu - a\big\} &\leq \exp^{-2a^2/n} \end{aligned} \tag{2.3}$$

Chernoff-Hoeffding Bound states that the probability that the empirical mean of random variables deviates from the expected value is bounded by exponential decay which depends on the deviation.

**Theorem 2.2.2.** [3] *(Bernstein Inequality) [5] Let $X_1, X_2, ..., X_K$ be independent random variables with zero means and variances $\sigma_1^2, \sigma_2^2, ..., \sigma_K^2$ such that for all integers $n > 2$,*

$$\mathbb{E}|X_i|^n \leq n! R^{n-2}\sigma_i^2/2 \text{ for all } i \in [K]$$

*for some constants $R > 0$. Then for all $t > 0$*

$$Pr\big\{\big|\sum_{i=1}^{K} X_i\big| \geq a\big\} \leq 2exp\big\{-\frac{a^2/2}{\sigma^2 + Rt}\big\} \tag{2.4}$$

*where $\sigma^2 = \sum_{i=1}^{K} \sigma_i^2$.*

[3] Throughout this thesis, for convenience of notation $[K]$ denotes for a consecutive positive integer list from 1 to K inclusively.

Similar to Chernoff-Hoeffding Bound, Bernstein Inequality bounds the empirical mean of zero-mean random variables within exponential decaying probability relative to their variances.

**Lemma 2.2.3.** *For $\kappa \neq 0$, the following inequality is true.*

$$\sum_{t=x+1}^{\infty} e^{-\kappa t} \leq \frac{1}{\kappa} e^{-\kappa x} \tag{2.5}$$

As continuous integral of $\int_{x+1}^{\infty} e^{-\kappa t} dt = \frac{1}{\kappa} e^{-k(x+1)} \leq \frac{1}{\kappa} e^{-kx}$ and similar conclusion holds for discrete cases as well.

### 2.2.3 Classical Approaches – Bernoulli Rewards

On top of problem definition in Section 2.2.1, we require that the given list of random variables $\langle X_1, X_2, ..., X_K \rangle$ be Bernoulli trials with parameters $\langle \theta_1, \theta_2, ..., \theta_K \rangle$. Drawing reward samples from $X_i$ is now equivalent to tossing the $i$th coin of head ($X_i = 1$) probability $\theta_i$.

#### Naive Epsilon-greedy Algorithm

The $\epsilon$-greedy [6] is the simplest yet widely used heuristic [7] [8] in the MAB problem class due to its ease of implementation. Given $K$ random variables, raw policy of $\epsilon$-greedy uniformly randomly selects $X_i$ from the candidate list with a typically small probability $\epsilon$ otherwise it selects the one with the highest empirical mean, so that at time step $t$, the probability of $X_i$ being chosen is $p_t(i)$.

$$p_t(i) = \begin{cases} \epsilon/K + 1 - \epsilon, & \text{if } \hat{\mu}_i(t-1) = max_i \hat{\mu}_i(t-1) \\ \epsilon/K, & \text{otherwise} \end{cases} \tag{2.6}$$

where $\hat{\mu}_i(t)$ denotes the empirical mean of $X_i$ at time $t$. At $t = 0$, when no random variable is selected yet, empirical means $\langle \hat{\mu}_1(0), \hat{\mu}_2(0), ..., \hat{\mu}_K(0) \rangle$ are often initialized as 0.

The $\epsilon$-greedy has only one parameter $\epsilon \in [0, 1]$ and it needs to be preset before the algorithm is used. Since $\epsilon$ is responsible for weighing the balance between exploration and exploitation, correct choice of $\epsilon$ value is key to

performance of the algorithm. A large $\epsilon$ brings more randomness to decision and $\epsilon = 1$ coerces the algorithm into pure exploration (random guessing). Conversely, $\epsilon = 0$ makes the algorithm constrict to conservative actions only with pure exploitation. If pure exploitation starts at $t = 0$, the $max_i\hat{\mu}_i(0)$ will always be picked up first and later decisions solely depend on empirical means computed from data. An apparent drawback of $\epsilon$-greedy with constant $\epsilon$ is the problem of infinite exploration. Suppose at time $t = T$ and $T$ is large enough so that enough amount of data has been collected; empirical means $\langle\hat{\mu}_1(T), \hat{\mu}_2(T), ..., \hat{\mu}_K(T)\rangle$ are so close to the true means $\langle\mu_1, \mu_2, ..., \mu_k\rangle$ that $max_i\hat{\mu}_i(T) = max_i\mu_i$. Whereas the optimal is revealed, exploration is no longer beneficial. Yet $\epsilon$-greedy does not stop here because there exists a fixed amount of exploration probability.

**Lemma 2.2.4.** *The $\epsilon$-greedy with a fixed $\epsilon$ has a non-sublinear expected cumulative regret: $\mathbb{E}[R_T] \sim \Theta(T)$.*

*Proof.* For convenience of notation, we let $X_1$ be the optimal arm by $\mu_1 = \mu^*$ without loss of generality. Assume $t$ is getting sufficiently large so that empirical means are close enough to true means for every arm. Then at a time step $t$, $\epsilon$-greedy policy has expected regret of $\mathbb{E}\big[\epsilon(\mu^* - \hat{\mu}_{i\neq 1}(t)) + (1 - \epsilon)(\mu^* - \hat{\mu}_{i=1}(t))\big]$. Then the expected cumulative regret is

$$\begin{aligned} R_T &= \mathbb{E}\big[\sum_{t=1}^{T} \epsilon(\mu^* - \hat{\mu}_{i\neq 1}(t)) + (1 - \epsilon)(\mu^* - \hat{\mu}_{i=1}(t))\big] \\ &= \epsilon\mathbb{E}\big[\sum_{t=1}^{T}(\mu^* - \hat{\mu}_{i\neq 1}(t))\big] + (1 - \epsilon)\sum_{t=1}^{T}\mathbb{E}\big[\mu^* - \hat{\mu}_{i=1}(t))\big] \\ &= \epsilon\mathbb{E}\big[\sum_{t=1}^{T}(\mu^* - \hat{\mu}_{i\neq 1}(t))\big] + (1 - \epsilon)\sum_{t=1}^{T}[\mathbb{E}\big[\mu^*\big] - \mathbb{E}\big[\hat{\mu}_{i=1}(t))\big]] \end{aligned} \tag{2.7}$$

By the above assumption $\mu_1 = \mu^*$, $\mathbb{E}\big[\mu^*\big] - \mathbb{E}\big[\hat{\mu}_{i=1}(t))\big] = 0$. Then let $X_{submax}$ be the second optimal arm so that $\mu_{submax} \geq \mu_{i\neq 1}$. Also let $\Delta = \mu^* - \mu_{submax}$

as a constant for any particular collection of arms.

$$\begin{aligned} R_T &= \epsilon \mathbb{E}\big[\sum_{t=1}^{T}(\mu^* - \hat{\mu}_{i\neq 1}(t))\big] \\ &= \epsilon \sum_{t=1}^{T} \mathbb{E}\big[(\mu^* - \hat{\mu}_{i\neq 1}(t))\big] \\ &= \epsilon \sum_{t=1}^{T}(\mu^* - \mu_{i\neq 1})\big] \\ &\geq \epsilon \sum_{t=1}^{T}(\mu^* - \mu_{submax})\big] = T\epsilon\Delta \text{ ($\Delta$ is constant)} \end{aligned} \tag{2.8}$$

□

Formula 2.8 shows that given constant $\epsilon$ cumulated expected reward grows at least in linear order, or $\mathbb{E}[R_T] \sim \Theta(T)$, thus proving Lemma 2.2.4.

**Epsilon-greedy Algorithm with Adaptive Rate**

Auer et al. [9] proposed an improved policy based on $\epsilon$-greedy by applying adaptive rules to parameter $\epsilon$. The basic idea is to introduce decay of the order $\frac{1}{t}$ so as to turn $\epsilon$ into a function of time $t$ instead of a constant. Auer et al. proved that the adaptive rate of exploration delivers sublinear expected cumulative regret for $k$-armed bandit problems with Bernoulli rewards for $k > 1$. Their improved $\epsilon$-greedy, namely $\epsilon_n$-greedy policy sets $\epsilon = \epsilon_t$ for time steps $t = 1, 2, ...$ by

$$\epsilon_t = min\big\{1, \frac{cK}{d^2 t}\big\}$$

where parameters $c > 0$ and $0 < d < 1$. Let $I_t$ be the index of the chosen arm and $[K]$ is the set of consecutive integers $1...K$. At time $t$, $I_t$ is randomly governed by $\epsilon_t$.

$$I_t = \begin{cases} \text{argmax}_{i\in[K]}\big\{\hat{\mu}_i(t-1)\big\} & \text{with probability } 1-\epsilon_t \\ i \in random\ from\ [K] & \text{with probability } \epsilon_t \end{cases} \tag{2.9}$$

The decision making policy is pretty much the same as non-adaptive $\epsilon$-greedy except that exploration factor gradually diminishes as time $t$ goes by. As $t$ gets large enough $\epsilon_n$-greedy policy eventually sticks to the arm with maximum empirical mean. This is a reasonable heuristic as we tend to trust our observation with increasing confidence when enough data is collected.

**Theorem 2.2.5.** *The $\epsilon_n$-greedy policy delivers a sublinear cumulative regret of order $R_T \sim o(T)$ for distributions $\langle X_1, X_2, ..., X_K \rangle$ with support in $[0, 1]$, if lower bound of $\mu^* - \mu_k$ is known for $k \in [K]$.*

*Proof.* Recall that $N_k(t)$ is the number of plays on arm $k$ at the point of time $t$ and $\hat{\mu}_k(t)$ is the empirical mean at arm $k$ by time $t$. Also define $N_k^R(t)$ to be the number of plays on arm $k$ *only because $X_k$ was randomly chosen.* Let $\hat{\mu}^*(t)$ be empirical mean from the optimal arm after time $t$. The algorithm has no awareness of which arm $\hat{\mu}^*(t)$ comes from. Then probability that arm $k$ is chosen is $Pr(I_t = k)$. Define $x_0 = \frac{1}{K} \sum_{t=1}^{T} \epsilon_t$.

$$\begin{aligned} Pr(I_t = k) &= Pr\big\{I_t = k \cap \hat{\mu}_k(t-1) \geq \mu^*\big\} + Pr\big\{I_t = k \cap \hat{\mu}_k(t-1) < \mu^*\big\} \\ &= Pr\big\{I_t = k \mid \hat{\mu}_k(t-1) \geq \mu^*\big\} Pr\big\{\hat{\mu}_k(t-1) \geq \mu^*\big\} \\ &\quad + Pr\big\{I_t = k \mid \hat{\mu}_k(t-1) < \mu^*\big\} Pr\big\{\hat{\mu}_k(t-1) < \mu^*\big\} \end{aligned} \tag{2.10}$$

Formula 2.6 gives conditional probability of arm $k$ being chosen when $X_k$ is either optimal or not. For the purpose of long term regret bounds, we may assume $t$ is large enough so that $\hat{\mu}^*(t) = max_i \hat{\mu}_i(t)$. With $\epsilon$ replaced with $\epsilon_t$ we plug both parts into Formula 2.10 so $Pr(I_t = k)$ gets simplified as

$$\begin{aligned} Pr(I_t = k) &= (1 - \epsilon_t + \frac{\epsilon_t}{K}) Pr\big\{\hat{\mu}_k(t-1) \geq \hat{\mu}^*(t-1)\big\} \\ &\quad + \frac{\epsilon_t}{K}(1 - Pr\big\{\hat{\mu}_k(t-1) \geq \hat{\mu}^*(t-1)\big\}) \\ &= \frac{\epsilon_t}{K} + (1 - \epsilon_t) Pr\big\{\hat{\mu}_k(t-1) \geq \hat{\mu}^*(t-1)\big\} \end{aligned} \tag{2.11}$$

Empirical means $\hat{\mu}_k(t-1)$ and $\hat{\mu}^*(t-1)$ in the following formula and 2.12 are represented as $\hat{\mu}_k$ and $\hat{\mu}^*$ for convenience of notation. Define $\Delta_k = \mu^* - \mu_k$. We now start to seek for an upper bound on $Pr(I_t = k)$.

$$Pr(I_t = k) \leq \frac{\epsilon_t}{K} + (1 - \epsilon_t) Pr\big\{\hat{\mu}_k \geq \hat{\mu}^*\big\}$$

and

$$\begin{aligned} Pr\{\hat{\mu}_k \geq \hat{\mu}^*\} &\leq Pr(\hat{\mu}_k \geq \mu_k + \frac{\Delta_k}{2} \ \cup \ \mu^* \leq \hat{\mu}^* - \frac{\Delta_k}{2}) \\ &\leq Pr(\hat{\mu}_k \geq \mu_k + \frac{\Delta_k}{2}) + Pr(\mu^* \leq \hat{\mu}^* - \frac{\Delta_k}{2}) \end{aligned} \tag{2.12}$$

since $\hat{\mu}_k \geq \mu_k + \frac{\Delta_k}{2}$ or $\mu^* \leq \hat{\mu}^* - \frac{\Delta_k}{2}$ is a necessary condition for $\hat{\mu}_k \geq \hat{\mu}^*$. The two terms on the right side of Formula 2.12 are not necessarily equal because distributions vary among arms. However, later steps will show that they share the same upper bound. Starting with the first term, at some time $T$ we have

$$\begin{aligned} &Pr\{\hat{\mu}_k(T) \geq \mu_k + \frac{\Delta_k}{2}\} \\ &= \sum_{t=1}^{T} Pr\{N_k(T) = t \cap \hat{\mu}_k(T) \geq \mu_k + \frac{\Delta_k}{2}\} \\ &= \sum_{t=1}^{T} Pr\{N_k(T) = t \mid \hat{\mu}_k(T) \geq \mu_k + \frac{\Delta_k}{2}\} Pr\{\hat{\mu}_k(T) \geq \mu_k + \frac{\Delta_k}{2}\} \end{aligned} \tag{2.13}$$

Chernoff-Hoeffding Bound says

$$Pr\{\hat{\mu}_k(T) \geq \mu_k + \frac{\Delta_k}{2}\} \leq e^{-\Delta_k^2 N_k(T)/2}$$

Continue with Formula 2.13.

$$
\begin{aligned}
&Pr\big\{\hat{\mu}_k(T) \geq \mu_k + \frac{\Delta_k}{2}\big\} \\
&\leq \sum_{t=1}^{T} Pr\big\{N_k(T) = t \mid \hat{\mu}_k(T) \geq \mu_k + \frac{\Delta_k}{2}\big\}\ e^{-\Delta_k^2 t/2} \\
&\leq \sum_{t=1}^{\lfloor x_0 \rfloor} Pr\big\{N_k(T) = t \mid \hat{\mu}_k(t) \geq \mu_k + \frac{\Delta_k}{2}\big\}\ e^{-\Delta_k^2 t/2} + \sum_{t=\lfloor x_0 \rfloor+1}^{T} e^{-\Delta_k^2 t/2} \\
&\leq \sum_{t=1}^{\lfloor x_0 \rfloor} Pr\big\{N_k(T) = t \mid \hat{\mu}_k(t) \geq \mu_k + \frac{\Delta_k}{2}\big\}\ e^{-\Delta_k^2 t/2} + \sum_{t=\lfloor x_0 \rfloor+1}^{\infty} e^{-\Delta_k^2 t/2} \\
&\leq \sum_{t=1}^{\lfloor x_0 \rfloor} Pr\big\{N_k(T) = t \mid \hat{\mu}_k(t) \geq \mu_k + \frac{\Delta_k}{2}\big\} + \sum_{t=\lfloor x_0 \rfloor+1}^{\infty} e^{-\Delta_k^2 t/2}
\end{aligned}
$$

By Lemma 2.2.3 (Formula 2.5), the second summation gets bounded.

(2.14)

$$
\begin{aligned}
&Pr\big\{\hat{\mu}_k(T) \geq \mu_k + \frac{\Delta_k}{2}\big\} \\
&\leq \sum_{t=1}^{\lfloor x_0 \rfloor} Pr\big\{N_k(T) = t \mid \hat{\mu}_k(t) \geq \mu_k + \frac{\Delta_k}{2}\big\} + \frac{2}{\Delta_k^2} e^{-\Delta_k^2 \lfloor x_0 \rfloor/2} \\
&\leq \sum_{t=1}^{\lfloor x_0 \rfloor} Pr\big\{N_k(T) = t\big\} + \frac{2}{\Delta_k^2} e^{-\Delta_k^2 \lfloor x_0 \rfloor/2} \\
&\text{As } N_k^R(T) \leq t \text{ is necessary condition for } N_k(T) = t \\
&\leq \sum_{t=1}^{\lfloor x_0 \rfloor} Pr\big\{N_k^R(T) \leq t\big\} + \frac{2}{\Delta_k^2} e^{-\Delta_k^2 \lfloor x_0 \rfloor/2} \\
&\leq x_0 Pr\big\{N_k^R(T) \leq x_0\big\} + \frac{2}{\Delta_k^2} e^{-\Delta_k^2 \lfloor x_0 \rfloor/2}
\end{aligned}
$$

(2.15)

$N_k^R(t)$ is essential a sum of Bernoulli trials during $t$ steps considering $X_k$ to be explored rather than exploited. Then

$$\mathbb{E}\big[N_k^R(T)\big] = \frac{1}{K} \sum_{t=1}^{T} \epsilon_t = 2x_0$$

and the sum of variances

$$Var\big[N_k^R(T)\big] = \sum_{t=1}^{T} \frac{\epsilon_t}{K}(1 - \frac{\epsilon_t}{K}) \leq \frac{1}{K}\sum_{t=1}^{T} \epsilon_t = 2x_0$$

$$\begin{aligned}
&Pr\big\{\hat{\mu}_k(T) \geq \mu_k + \frac{\Delta_k}{2}\big\} \\
&\leq x_0 Pr\big\{N_k^R(T) - 2x_0 \leq -x_0\big\} + \frac{2}{\Delta_k^2} e^{-\Delta_k^2 \lfloor x_0 \rfloor/2} \\
&\leq x_0 Pr\big\{|N_k^R(T) - 2x_0| \geq x_0\big\} + \frac{2}{\Delta_k^2} e^{-\Delta_k^2 \lfloor x_0 \rfloor/2} \\
&\leq x_0 e^{-x_0/5} + \frac{2}{\Delta_k^2} e^{-\Delta_k^2 \lfloor x_0 \rfloor/2}
\end{aligned} \tag{2.16}$$

Bernstein Inequality (Theorem 2.4) is applicable here since $N_k^R(T) - 2x_0$ has a zero mean. The only remaining work for this proof is to lower bound $x_0$. Recall that $\epsilon_t = min\big\{1, \frac{cK}{d^2 t}\big\}$ so only when $t \leq cK/d^2$, $\epsilon_t \geq 1$. Let $T' = cK/d^2$.

$$\begin{aligned}
x_0 &= \frac{1}{2K}\sum_{t=1}^{T'} \epsilon_t + \frac{1}{2K}\sum_{t=T'+1}^{T} \epsilon_t \\
&= \frac{1}{2K} T' + \frac{1}{2K}\sum_{t=T'+1}^{T} \epsilon_t \\
&\geq \frac{T'}{2K} + \frac{c}{d^2} ln \frac{T}{T'} \\
&\geq \frac{c}{d^2} ln\big\{(exp\{\frac{1}{2}\frac{T'd^2}{cK}\})\frac{T}{T'}\big\} \\
&\geq \frac{c}{d^2} ln\big\{exp\{\frac{1}{2}\frac{T'd^2}{cK}\}\frac{T}{T'}\big\} \qquad\qquad (\text{note } \frac{T'd^2}{cK} \geq 1) \\
&\geq \frac{c}{d^2} ln\big\{\frac{Td^2}{cK} e^{1/2}\big\}
\end{aligned} \tag{2.17}$$

Back to Formula 2.11 we are trying to bound for this proof, with 2.17

and 2.16

$$
\begin{aligned}
Pr(I_t = k) &\leq \frac{\epsilon_{t-1}}{K} + (1 - \epsilon_{t-1}) Pr\big\{\hat{\mu}_k(t-1) \geq \hat{\mu}^*\big\} \\
&\leq \frac{\epsilon_{t-1}}{K} + 2x_0 e^{-x_0/5} + \frac{4}{\Delta_k^2} e^{-\Delta_k^2 \lfloor x_0 \rfloor / 2} \\
&\leq \frac{c}{td^2} + \frac{2c}{d^2} (\ln \frac{(t-1)d^2 e^{1/2}}{cK}) (\frac{2c}{(t-1)d^2 e^{-1/2}})^{c/5d^2} + \frac{4}{\Delta_k^2} (\frac{cK}{(t-1)d^2\sqrt{e}})^{\frac{c\Delta_k^2}{2d^2}}
\end{aligned}
\tag{2.18}
$$

The last term in Formula 2.18 differs from the original conclusion given by Auer et al. since we impose no further assumption on $d$ as their original proof did. For the last three terms in Formula 2.18, the first indicates a clear bound of order $O(1/t)$ and the second is $o(1/t)$ for $c > 5$. For the third term to be in $o(1/t)$, a exponential larger than 1 is required, i.e., $c\Delta_k^2 > 2d^2$, or equivalently a lower bound on $\Delta_k$ is known. In this case the maximum probability of choosing a suboptimal arm under $\epsilon_n$-policy after $t-1$ trials in its worst case is $O(1/t)$ so expected cumulative regret is bounded by $O(log(t))$ thereby Theorem 2.2.5 is proved.

□

**Upper Confidence Bound (UCB1)**

Another classical policy worth examining is called UCB1 proposed by the same authors [9] of $\epsilon_n$-greedy. As a simplistic yet popularly adopted MAB policy, it differs from $\epsilon$-greedy approaches which introduces randomness for balancing exploration factor against exploitation. UCB1 is an "optimism" based tradeoff algorithm instead of explicitly manipulating stepwise decisions during the MAB experiment. The algorithm requires no input parameter but measures how optimistic we are for the expected reward from every arm with some metric called *upper confidence bound* (hence the acronym UCB). The intuition about upper confidence bound is our uncertainty of the true mean of some distribution. The more data we collect from a specific arm, the less optimistic (or more confident) we are about its reward upper bound because an increasing amount of data reinforces our certainty on its empirical mean. To apply UCB1 policy to the $K$-armed bandit problem, we sequentially select every arm once as initialization step. Starting from time $t = K + 1$, UCB1

always acts on arm $X_j$ such that

$$j = \underset{j \in [K]}{\operatorname{argmax}} \big\{ \hat{\mu}_j(t-1) + \frac{2 \ln t}{N_k(t-1)} \big\}$$

For arms that are rarely selected, $N_k(t-1)$ has a small value so a larger confidence interval could compensate its empirical mean. The design of UCB1 mainly encourages actions on arms that are not often explored. Unconditionally, UCB1 policy has a $O(lnT)$ expected cumulative regret bound.

**Theorem 2.2.6.** *The worst case expected cumulative regret under UCB1 policy is logarithmic for distributions* $\langle X_1, X_2, ..., X_K \rangle$ *with support in* $[0, 1]$. $R_T \sim O(lnT)$.

*Proof.* Let $\Delta_k = \mu^* - \mu_k$. Define $c(t, n) = \sqrt{2 \ln t / n}$ and $\mathbb{1}\big[event\big]$ as an indicator function for an event. $\mathbb{1}\big[event\big] = 1$ only if the event is true otherwise 0. As previously defined $I_t$ is the index of chosen arm at time $t$. If the expected number of plays on $X_k$ up to time $t$ is $\mathbb{E}[N_k(t)]$, then the expected cmumulative regret can be bounded with $\sum_k \Delta_k \mathbb{E}[N_k(t)]$. So the proof is equivalent to finding the worst case $\mathbb{E}[N_k(t)]$. Since for the first $K$ rounds every arm gets selected for once, by definition,

$$N_k(T) = 1 + \sum_{t=K+1}^{T} \mathbb{1}\big[I_t = k\big]$$

Let $\ell$ be some positive integer, then

$$N_k(T) \leq \ell + \sum_{t=K+1}^{T} \mathbb{1}\big[I_t = k \cap N_k(t-1) \geq \ell\big]$$

Then

$$\begin{aligned} N_k(T) &\leq \ell + \sum_{t=K+1}^{T} \mathbb{1}\big[I_t = k \cap N_k(t-1) \geq \ell\big] \\ &= \ell + \sum_{t=K+1}^{T} \mathbb{1}\big[\mu_k(t-1) + c(t-1, N_k(t-1)) \geq \\ &\qquad \mu^*(t-1) + c(t-1, N^*(t-1)) \cap N_k(t-1) \geq \ell\big] \end{aligned} \tag{2.19}$$

For $\hat{\mu}_k(t-1) + c(t-1, N_k(t-1)) \geq \hat{\mu}^*(t-1) + c(t-1, N^*(t-1))$ to be true at time $t$, the following necessary condition must be met.

$$min_{0<t_0<t}\big[\hat{\mu}^*(t-1)+c(t-1, N^*(t_0-1))\big] \leq max_{0<t_1<t}\big[\hat{\mu}_k(t-1)+c(t-1, N_k(t_1-1))\big]$$

for some $t_0, t_1$. If $N_k(t-1) \geq \ell$ is further required,

$$min_{0<t_0<t}\big[\hat{\mu}^*(t-1)+c(t-1, N^*(t_0-1))\big] \leq max_{\ell\leq t_1<t}\big[\hat{\mu}_k(t-1)+c(t-1, N_k(t_1-1))\big]$$

has to be true. Then a crude relaxation can be imposed to the inequality 2.19.

$$\begin{aligned}
N_k(T) \leq \ell + \sum_{t=K+1}^{T} \mathbb{1}\big[&min_{0<t_0<t}\big[\hat{\mu}^*(t-1) + c(t-1, N^*(t-1))\big] \leq \\
&max_{\ell<t_1<t}\big[\hat{\mu}_k(t-1) + c(t-1, N_k(t_1-1))\big]\big] \\
&\leq \sum_{t=1}^{T}\sum_{t_0=1}^{t-1}\sum_{t_1=\ell}^{t-1} \mathbb{1}\big[\hat{\mu}^*(t-1) + c(t-1, N^*(t_0-1)) \leq \\
&\hat{\mu}_k(t-1) + c(t-1, N_k(t_1-1))\big] \\
&\leq \sum_{t=1}^{\infty}\sum_{t_0=1}^{t-1}\sum_{t_1=\ell}^{t-1} \mathbb{1}\big[\hat{\mu}^*(t) + c(t, N^*(t_0-1)) \leq \hat{\mu}_k(t) + c(t, N_k(t_1-1))\big]
\end{aligned} \tag{2.20}$$

In order that $\hat{\mu}^*(t) + c(t, N^*(t_0-1)) \leq \hat{\mu}_k(t) + c(t, N_k(t_1-1))$, at least one of the following three claims has to be true.

$$\hat{\mu}^*(t) \leq \mu^* - c(t, N^*(t_0-1)) \tag{2.21}$$

$$\hat{\mu}_k(t) \geq \mu_k + c(t, N_k(t_1-1)) \tag{2.22}$$

$$\mu^* - \mu_k < 2c(t, N_k(t_1-1)) \tag{2.23}$$

It is easy to show that if all the above three claims were false,

$$\begin{aligned}
2c(t, N_k(t_1-1)) &\leq \mu^* - \mu_k \\
&< \hat{\mu}^*(t) - \hat{\mu}_k(t) + c(t-1, N_k(t_1-1) + c(t, N^*(t_0-1))
\end{aligned}$$

Then

$$\hat{\mu}^*(t) - \hat{\mu}_k(t) > c(t-1, N_k(t_1-1) - c(t, N^*(t_0-1))$$

contradicting to the event of premise. □

We apply Chernoff-Hoeffding bound to Claim 2.21 and 2.22.

$$Pr\{\hat{\mu}^*(t) \leq \mu^* - c(t, N^*(t_0-1))\} \leq \exp\{-4\ln\frac{t}{N^*(t_0-1)}N^*(t_0-1)\} = t^{-4}$$

$$Pr\{\hat{\mu}_k(t) \geq \mu_k + c(t, N_k(t_1-1))\} \leq \exp\{-4\ln\frac{t}{N^*(t_1-1)}N^*(t_1-1)\} = t^{-4}$$

Setting $\ell = \lceil\frac{8\ln t}{\Delta_i^2}\rceil$ makes Claim 2.23 constantly false because 2.19 requires $N_k(t_1-1) \geq \ell$ which yields

$$\mu^* - \mu_k - 2c(t, N_k(t_1-1)) = \mu^* - \mu_k - 2\sqrt{\frac{2\ln t}{N_k(t_1-1)}} \geq \mu^* - \mu_k - \Delta_k = 0$$

So Claim 2.23 always fails. Apply union bound to Formula 2.20.

$$\begin{aligned}
\mathbb{E}[N_k(t)] = \mathbb{E}[N_k(T)] &\leq \lceil\frac{8\ln t}{\Delta_i^2}\rceil + \sum_{t=1}^{\infty}\sum_{t_0=1}^{t-1}\sum_{t_1=\ell}^{t-1} \\
&\quad \big(Pr\{\hat{\mu}^*(t) \leq \mu^* - c(t, N^*(t_0-1))\} + Pr\{\hat{\mu}_k(t) \geq \mu_k + c(t, N_k(t_1-1))\}\big) \\
&\leq \lceil\frac{8\ln t}{\Delta_i^2}\rceil + \sum_{t=1}^{\infty}\sum_{t_0=1}^{t}\sum_{t_1=1}^{t}\left(t^{-4}+t^{-4}\right) \\
&\leq \frac{8\ln t}{\Delta_i^2} + 1 + \frac{\pi}{3}
\end{aligned} \tag{2.24}$$

We thus conclude the proof.

### 2.2.4 Classical Approaches – Gaussian Rewards

There exists a variant of UCB1 named UCB1-NORMAL proposed in the same paper [9] and it proves to hold a $O(lnT)$ expected cumulative regret for bandit problems with normally distributed rewards. More recent works provide more general modified versions of UCB such as Bayes-UCB [3] which is designed to more efficiently deal with normal rewards; UCB-V [10] which does not assume distribution category of rewards with bounded support; and GP-UCB that is specially optimized for Gaussian process and assumes no support bounds. The next section will make in-depth discussion on GP-UCB and its relation to our proposed model. For the final part of this section,

we instead introduce a non-UCB based simple approach called Thompson sampling commonly used as a dithering technique for optimizing bandits with normally distributed rewards.

### Thompson Sampling

Back to the general $K$-armed bandit problem setting in Section 2.2.1, assume bandit reward $X_k$ has normal distribution and the reward list $\boldsymbol{r_k} = \langle r_k^{(1)}, r_k^{(2)}, ..., r_k^{(n)} \rangle$ contains all the observations collected from $X_k$ so far. We are interested in discovering the underlying reward distribution $Pr(\mu_k \mid \boldsymbol{r_k})$ based on currently available data using Bayesian paradigm.

$$Pr(\mu_k \mid \boldsymbol{r_k}) \propto Pr(\boldsymbol{r_k} \mid \mu_k) Pr(\mu_k)$$

As the conjugate prior of normal distribution is normal as well given pre-assumed variance, we expect the prior probability $Pr(\mu_k) \sim \mathcal{N}(\mu'_k, 1/\tau'_k)$. Let $\tau_k = 1/\sigma_k$ be the precision of normal distribution of $X_k$, we have the following posterior distribution.

$$\begin{aligned}
&Pr(\mu_k \mid \boldsymbol{r_k}) \\
&\propto \big[ \sum_i \frac{1}{\sqrt{2\pi/\tau_k}} \exp\big\{ -\frac{\tau_k}{2}(\mu_k - r_k^{(i)})^2 \big\} \big] \frac{1}{\sqrt{2\pi/\tau'_k}} \exp\big\{ -\frac{\tau_k}{2}(\mu_k - \mu'_k)^2 \big\} \\
&\propto \exp\big\{ -\frac{\tau'_k + \sum_i \tau_k}{2}\mu_k^2 + (\tau_k \sum_i r_k^{(i)} + \tau'_k \mu'_k)\mu_k + C \big\} \\
&\propto \exp\big\{ -\frac{\tau'_k + n\tau_k}{2} \big[ \mu_k - \frac{\tau_k \sum_i r_k^{(i)} + \tau'_k \mu'_k}{\tau'_k + n\tau_k} \big]^2 \big\}
\end{aligned} \tag{2.25}$$

After dropping constants irrelevant to $\mu_k$, Formula 2.25 describes the posterior distribution of single-arm rewards. In practice, $\tau_k$, $\tau'_k$ are commonly initialized as 1 and $\mu'_k$ starts with 0. If we prefer to make greedy actions by choosing the arm with the maximum mean of posterior distribution resembling the Bayesian A/B test illustrated in Section 2.1 we may simply stick to Formula 2.25 for computing the means. Greedy actions can practically be problematic because it fails to account for candidate arms with high uncertainty (variance). Consequently the sequential decision process gets inclined to exploit more certain candidate distributions and refuse to explore truly the optimal action.

Thompson sampling [11] addresses the greedy issue by perturbing estimated parameters with randomness. We know from Formula 2.25 that posterior probability of $\mu_k$ with observations is normal.

$$Pr(\mu_k \mid \boldsymbol{r_k}) \sim \mathcal{N}(\frac{\tau_k \sum_i r_k^{(i)} + \tau'_k \mu'_k}{\tau'_k + n\tau_k}, \frac{1}{\tau'_k + n\tau_k}) \tag{2.26}$$

For every arm we sample an empirical parameter $\hat{\mu}_k$ from $Pr(\mu_k \mid \boldsymbol{r_k})$ and play the arm indexed $I$ with the largest sample.

$$I = \underset{k \in [K]}{\operatorname{argmax}} \hat{\mu}_k \sim Pr(\mu_k \mid \boldsymbol{r_k})$$

Effectiveness of Thompson sampling comes from uncertainty of less played arms because such distributions are more likely to produce samples surpassing arms that were exhaustively explored. Initialization of $\tau'_k$ and $\tau_k$ in ( 2.26) can be overridden to alternative values to adjust how fast uncertainty of an arm decreases as its observation count $n$ grows. More in-depth study of Thompson sampling will be discussed with experiments in later chapters.

## 2.3 Gaussian Process Regression

This section concisely elaborates Gaussian Process regression, a non-parametric supervised learning model and thoroughly discusses GP-UCB [12], a GP based Bayesian optimization method for stochastic bandit problems with sublinear regret bounds and the optimal action guaranteed, since the proposed model in this thesis is closely bonded with the core idea of GP-UCB, as is shown in the next chapter.

### 2.3.1 Introduction to Gaussian Process Regression

A Gaussian Process can simplistically be defined as a collection of correlated random variables, any finite subset of which has a multivariate Gaussian distribution [13]. A Gaussian process is completely specified by a mean function $\mu(x)$ and a covariance/kernel function $k(x, x')$ as $\mathcal{GP}(\mu(x), k(x, x'))$ where $x \in D$. $D$ is the index space $R^d$ and can be in one or higher dimensions ($d \geq 1$).

$$\begin{aligned} \mu(x) &= \mathbb{E}[f(x)] \\ k(x, x') &= \mathbb{E}[(f(x) - \mu(x))(f(x') - \mu(x'))] \end{aligned} \tag{2.27}$$

In 2.27, $f : D \to \mathbb{R}$ is a stochastic function such that $f(x)$ for any $x \in D$ is a random variable of normal distribution. In machine learning context, $f$ is the objective function we try to optimize and expect $\mathbb{E}[f(x)]$ to predict "ground truth" at location $x$. Hence a Gaussian process is often interpreted as a Gaussian distribution over functions.

$$f(x) \sim \mathcal{GP}(\mu(x), k(x, x'))$$

**Supervised Learning with Gaussian Process**

Let $\boldsymbol{x} = [x_1, ..., x_n]$ be the list of available features in training data and $\boldsymbol{y} = [y_1, ..., y_n]$ be training labels. Features of the test data is denoted as $\boldsymbol{x_*} = [x_1^*, ..., x_n^*]$. Define random variable lists $\boldsymbol{f} = [f(x_1), ..., f(x_n)]^T$ and $\boldsymbol{f_*} = [f(x_1^*), ..., f(x_n^*)]^T$. For simplicity, the means of prior distributions are assumed to be 0 so that $\boldsymbol{f}$ combined with $\boldsymbol{f_*}$ has a joint Gaussian prior.

$$\begin{bmatrix} \boldsymbol{f} \\ \boldsymbol{f_*} \end{bmatrix} \sim \mathcal{N}\big(\boldsymbol{0}, \begin{bmatrix} K(\boldsymbol{x}, \boldsymbol{x}) & K(\boldsymbol{x}, \boldsymbol{x_*}) \\ K(\boldsymbol{x_*}, \boldsymbol{x}) & K(\boldsymbol{x_*}, \boldsymbol{x_*}) \end{bmatrix}\big)$$

$K(\boldsymbol{x}, \boldsymbol{x})$ is the $n \times n$ covariance matrix calculated from the training data. The submatrices $K(\boldsymbol{x}, \boldsymbol{x_*})$ and $K(\boldsymbol{x_*}, \boldsymbol{x})$ are $n \times n_*$ and $n_* \times n$ covariance matrices whose entries $K_{ij}$, $K_{ji}$ express $k(x_i, x_j^*)$ and $k(x_j^*, x_i)$. As kernel functions $k$ are in most cases symmetric, $K_{ij} = K_{ji}$ is true. Similarly $K(\boldsymbol{x_*}, \boldsymbol{x_*})$ is a $n_* \times n_*$ square matrix representing test data covariance.

Given training data $\boldsymbol{x}$, training labels $\boldsymbol{y}$ and test features $\boldsymbol{x}_*$, prediction on $\boldsymbol{x_*}$ is given by the following posterior distribution.

$$\begin{aligned} &Pr(\boldsymbol{f}_* \mid \boldsymbol{x}_*, \boldsymbol{x}, \boldsymbol{y}) \sim \\ &\mathcal{N}\big(K(\boldsymbol{x_*}, \boldsymbol{x})K(\boldsymbol{x}, \boldsymbol{x})^{-1}\boldsymbol{y}, K(\boldsymbol{x_*}, \boldsymbol{x_*}) - K(\boldsymbol{x_*}, \boldsymbol{x})K(\boldsymbol{x}, \boldsymbol{x})^{-1}K(\boldsymbol{x}, \boldsymbol{x_*})\big) \end{aligned} \tag{2.28}$$

In real world applications, it is more practical to estimate some noise in the training data, i.e., labels $\boldsymbol{y}$ are not necessarily the "ground truth" values of $\boldsymbol{f}$ but $y_i = f(x_i) + \epsilon$ where $\epsilon \sim \mathcal{N}(0, \sigma^2)$. Assuming noise $\epsilon$ is i.i.d for $i = 1, ..., n$ the prior distribution of the stacked random vector now becomes

$$\begin{bmatrix} \boldsymbol{f} \\ \boldsymbol{f_*} \end{bmatrix} \sim \mathcal{N}\big(\boldsymbol{0}, \begin{bmatrix} K(\boldsymbol{x}, \boldsymbol{x}) + \sigma^2 \boldsymbol{I} & K(\boldsymbol{x}, \boldsymbol{x_*}) \\ K(\boldsymbol{x_*}, \boldsymbol{x}) & K(\boldsymbol{x_*}, \boldsymbol{x_*}) \end{bmatrix}\big)$$

Prediction given by noisy Gaussian process regression follows distribution below.

$$\begin{aligned} Pr(\boldsymbol{f}_* \mid \boldsymbol{x}_*, \boldsymbol{x}, \boldsymbol{y}) &\sim \mathcal{N}\big(\mu(\boldsymbol{f}_*), cov(\boldsymbol{f}_*)\big) \\ \text{where } \mu(\boldsymbol{f}_*) &= K(\boldsymbol{x_*}, \boldsymbol{x})[K(\boldsymbol{x}, \boldsymbol{x}) + \sigma^2 \boldsymbol{I}]^{-1} \boldsymbol{y} \\ cov(\boldsymbol{f}_*) &= K(\boldsymbol{x_*}, \boldsymbol{x_*}) - K(\boldsymbol{x_*}, \boldsymbol{x})[K(\boldsymbol{x}, \boldsymbol{x}) + \sigma^2 \boldsymbol{I}]^{-1} K(\boldsymbol{x}, \boldsymbol{x_*}) \end{aligned} \tag{2.29}$$

There are situations where it is necessary to assume training data is noise free even if $\epsilon$ exists. For example, when estimating a uniform hyperparameter $\sigma^2$ from training data is difficult and $\epsilon$ is significantly small, noise free regression can still produce satisfactory prediction.

### 2.3.2 GP-UCB Algorithm – Regression Under Bandit Setting

Section 2.3.1 defines a general Gaussian process in what is called the "function space view" because all the random variables are considered samples from a black box function $f : D \to \mathbb{R}$. Placed under bandit problem setting, a Gaussian process can be reformulated into a sequential optimization problem. In each round $t$ we make a decision at location $x_t \in D$ as candidate arms are indexed by action space $D$. For every sample $x_t$ we collected, the posterior of GP is updated. Here two goals are of our concern. First we want to minimize the total regret $R_T$ during the process as typical required by MAB problems. $R_T = \sum_{t=1}^{T} f(x^*) - f(x_t)$ and $x^*$ locates the global maxima of $f$. Second true value of $x^*$ can be eventually found such that an algorithm possesses the property $\lim_{T\to\infty} R_T/T = 0$.

Srinivas et al. [12] proposed a variant of upper confidence bound algorithm, Gaussian process upper confidence bound (GP-UCB) which holds the asymptotically zero-regret property and provably achieves novel bounds of cumulative regret. GP-UCB follows a very straightforward rule by choosing $x_t$ at every step meanwhile collecting a new data sample $y_t$.

Decision $x_t$ relies on both current GP status $sigma$, $\mu$ and appropriate constants $\beta_t$. Noise $\epsilon$ has to be estimated as some i.i.d of $\mathcal{N}(0, \sigma)$ before any

**Algorithm 1** GP-UCB [12]

init GP prior $\boldsymbol{\mu}$, $\boldsymbol{\sigma}$, $K$
**for** $t$ in step 1...T **do**
  choose $x_t = \text{argmax}_{x \in D}\, \mu_{t-1}(x) + \sqrt{\beta_t}\sigma_{t-1}(x)$
  sample $y_t = f(x_t) + \epsilon$
  update GP posterior
**end for**

experiments. In this thesis we focus on $\beta_t$ defined as below.[4]

$$\beta_t = 2 \ln \big[ \frac{|D| t^2 \pi^2}{6\delta} \big]$$

where constant $\delta \in (0, 1)$.

GP-UCB appears to be less intuitive than UCB1 whereas $\beta_t$ does not depend on locations $x$. As a result less explored locations do not secure fairness of being chosen with higher probability even though they are likely to be highly uncertain (large variances). From a high level view it is because that GP-UCB intends more quickly discover the optimal local $x^*$ by avoiding globally exploring the action space; and that for the sake of maximizing $f(x)$ a lot of redundant exploration in UCB1 could have been removed. The improvement of GP-UCB lies within its less greedy decision strategy whose regret bound is bounded by what is called maximum information gain during $T$ rounds.

### 2.3.3 GP-UCB – Information Gain and Regret Bounds

#### Information Gain of GP regression

Information gain(IG) [14] is used to describe the amount of mutual information shared between two probability distributions. High information gain generally indicates that revealing one distribution $X$ fortifies our confidence

[4] In the original works, Srinivas et al. proposed three different rules depending on context. The first rule discusses general finite discrete space $D$. The second rule applies to compact and convex space with assumption on kernels. The third warrants regret bound given that $f$ lies within Hilbert space. Only the first rule is mentioned here because latter cases are not closely relevant to the core problem of this thesis.

in the other $Y$. IG from $Y$ to $X$ is defined as the drop of entropy after $Y$ is revealed: $I(X;Y) = H(X) - H(X \mid Y)$. It is easy to show that

$$H(X) - H(X \mid Y) = H(Y) - H(Y \mid X) = \sum_{x,y} \frac{Pr(X = x, Y = y)}{Pr(X = x)Pr(Y = y)}$$

so $I(X;Y) = I(Y;X)$ is true. Back to the Gaussian process, regression is essentially reducing the entropy of the ground truth $f$ in our belief through sequentially collecting $y_t = f(x_t) + \epsilon$. To quantify the additional amount of useful information a training set $\langle \boldsymbol{x}_A, \boldsymbol{y}_A \rangle$ of size $|A|$ delivers, we define the IG of the training set as

$$I(\boldsymbol{f}_A; \boldsymbol{y}_A) = H(\boldsymbol{f}_A) - H(\boldsymbol{f}_A \mid \boldsymbol{y}_A)$$

where $A \subset D$ and $\boldsymbol{f}_A = [f(x)]_{x \in A}$ which are ground truth values the algorithm has no access to. Alternatively,

$$I(\boldsymbol{f}_A; \boldsymbol{y}_A) = I(\boldsymbol{y}_A; \boldsymbol{f}_A) = H(\boldsymbol{y}_A) - H(\boldsymbol{y}_A \mid \boldsymbol{f}_A) \tag{2.30}$$

For multivariate Gaussian, $H(\mathcal{N}(\mu, \Sigma)) = \frac{1}{2} \ln |2\pi e \Sigma|$. As $\boldsymbol{y}_A \sim \mathcal{N}(\boldsymbol{\mu}_A, \boldsymbol{K}_A + \sigma^2 \boldsymbol{I})$, where $\boldsymbol{K}_A$ is the kernal matrix of entries $k(x, x')_{x,x' \in A}$ and $\boldsymbol{\mu}_A$ does not matter, $H(\boldsymbol{y}_A) = \frac{1}{2} \ln |2\pi e(\boldsymbol{K}_A + \sigma^2 \boldsymbol{I})|$. Similarly, if $\boldsymbol{f}_A$ were to be revealed, the only uncertainty on $\boldsymbol{y}_A$ remains as noise so $H(\boldsymbol{y}_A \mid \boldsymbol{f}_A) = \frac{1}{2} \ln |2\pi e \sigma^2 \boldsymbol{I}|$. Rewriting Formula 2.30 computes information gain by training data $\langle \boldsymbol{x}_A, \boldsymbol{y}_A \rangle$.

$$I(\boldsymbol{f}_A; \boldsymbol{y}_A) = \frac{1}{2} \ln |\boldsymbol{I} + \sigma^{-2} \boldsymbol{K}_A| \tag{2.31}$$

Formula 2.30 makes it tempting for us to figure out a set $A \subset D$, sequentially and additively formed by a training set $\langle \boldsymbol{x}_A, \boldsymbol{y}_A \rangle$ such that information gain towards $f_A$ is maximized. Unfortunately finding the IG maximizer subject to $T$ steps ($A \subset D, |A| \leq T$) is NP-complete according to Krause & Guestrin [15], who further proposed an approximation algorithm that simply selects data $x_t$ in addition to $A_{t-1}$ with maximized marginal increase of IG at every step. Let $I(\boldsymbol{f}_A; \boldsymbol{y}_A) = F(A)$ then

$$x_t = \underset{x \in D}{\operatorname{argmax}} \left[ F(A_{t-1} \cup \{x_t\}) - F(A_{t-1}) \right]$$

The subset $A$ selected by this greedy approach after $T$ steps guarantees [15] a constant fraction of optimal IG granted by a truly optimal $A$.

$$F(A_T) \geq (1 - 1/e) \max_{A \subset D, |A| \leq T} F(A)$$

It is because $F(A)$ satisfies "diminishing return"" property called *submodularity* so that marginal increase of $A$ is non-increasing.

$$F(A \cup \boldsymbol{x}) - F(A) \geq F(A' \cup \boldsymbol{x}) - F(A')$$

holds for all $A \subset A' \subset D$, $\boldsymbol{x} \neq A$. It is proved by Nemhauser et al. [16] that greedy approximation is at most a constant factor worse than the optimal answer. In the case of Gaussian process regression, this approximation is equivalent to picking $x_t$ with the largest variance.

$$x_t = \underset{x \in D}{\operatorname{argmax}}\, \sigma_{t-1}(x)$$

As previously mentioned, GP-UCB algorithm tends to stay more conservative against global exploration than greedy approximation, yet able to achieve total regret bounded by the maximum information gain which does not need to be precisely quantified in its algorithm.

### Regret Bound of GP-UCB

With the problem setting in Section 2.3.2 running GP-UCB for $T$ rounds makes up a sample set $A$ of size $T$. If $A$ is an optimal set that maximizes $I(\boldsymbol{f}_A; \boldsymbol{y}_A)$ then the IG it achieved is defined as $\gamma_T = max_{A \subset D, |A|=T} I(\boldsymbol{f}_A; \boldsymbol{y}_A)$. Without finding such $A$, GP-UCB obtains a total regret bounded by $\gamma_T$.

**Theorem 2.3.1.** *Running GP-UCB for a sample $f$ with 0 GP mean and covariance function $k(x, x') \leq 1$, the sum of regret $R_T$ is bounded by $O^*(T\gamma_T ln|D|)$ with high probability. Precisely,*

$$Pr\big\{R_T \leq \sqrt{C_1 T \beta_T \gamma_T} \;\; \forall T \geq 1\big\} \geq 1 - \delta$$

*where $C_1 = 8/log(1 + \sigma^{-2})$ and $\delta \in (0, 1)$.*

Srinivas et al. provided four lemmas listed below in order to prove Theorem 2.3.1 and those lemmas also give sound explanation to the $\beta_t$ value of choice.

**Lemma 2.3.2.** *$Pr\big\{|f(x) - \mu_{t-1}(x)| \leq \sqrt{\beta_t}\sigma_{t-1}(x)\big\} \geq 1 - \delta$, $\forall x \in D$ $\forall t \geq 1$ holds for $\delta \in (0, 1)$ and $\beta_t = 2\ln(|D|\pi_t/\delta)$ if $\sum_t^\infty 1/\pi_t = 1$*

*Proof.* Since $f(x)$ is the sample function from a GP, $f(x) \sim \mathcal{N}(\mu_{t-1}(x), \sigma_{t-1}(x))$. Let $r = (f(x) - \mu_{t-1}(x))/(\sigma_{t-1}(x))$ making $r$ standard normal distribution. Pick $c > 0$ and $Pr\{r > c\} = \int_c^\infty \varphi(r)dr$; $\varphi(r)$ is standard normal density.

$$\begin{aligned}
Pr\{r > c\} &= \frac{1}{\sqrt{2\pi}} \int_c \exp[-\frac{r^2}{2}]dr = \exp[\frac{c^2}{2}]\frac{1}{\sqrt{2\pi}} \int_c \exp[\frac{c^2 - r^2}{2}]dr \\
&= \exp[\frac{-c^2}{2}]\frac{1}{\sqrt{2\pi}} \int_c \exp[-(r-c)^2/2] \exp[-c(r-c)]dr \\
&\text{for } 0 < c < r, \exp[-c(r-c)] < 1 \\
Pr\{r > c\} &\leq \exp[\frac{-c^2}{2}]\frac{1}{\sqrt{2\pi}} \int_c \exp[-(r-c)^2/2]dr \\
&= \exp[\frac{-c^2}{2}]\frac{1}{\sqrt{2\pi}} \int_0 \exp[-r^2/2]dr \\
&= \exp[\frac{-c^2}{2}]Pr\{r > 0\} = \frac{1}{2}\exp[\frac{c^2}{2}]
\end{aligned} \tag{2.32}$$

The two-tail probability is $Pr\{|r| > c\} \leq \exp(-\frac{c^2}{2})$. Now plug in $r$ value and let $c = \sqrt{\beta_t}$.

$$\begin{aligned}
Pr\{|\frac{(f(x) - \mu_{t-1}(x))}{\sigma_{t-1}(x)}| > \sqrt{\beta_t}\} &\leq \exp(-\beta_t/2) \\
\Rightarrow Pr\{|f(x) - \mu_{t-1}(x)| > \sigma_{t-1}(x)\sqrt{\beta_t}\} &\leq \exp(-\beta_t/2)
\end{aligned} \tag{2.33}$$

Apply (2.33) to all $x \in D$,

$$Pr\{|f(x) - \mu_{t-1}(x)| \leq \sigma_{t-1}(x)\sqrt{\beta_t},\ \forall x \in D\} \geq 1 - |D|\exp(-\beta_t/2) = 1 - \frac{\sigma}{\pi_t}$$

□

A wise choice of $\pi_t$ is $\pi^2 t^2/6$ because this makes $\sum_{t=1}^\infty 1/\pi_t = 1$ and $\beta_t = 2\ln\frac{|D|\pi^2 t^2}{6\delta}$, which explains the selected value of appropriate constant used in GP-UCB.

**Lemma 2.3.3.** *For $t \geq 1$, $|f(x) - \mu_{t-1}(x)| \leq \sqrt{\beta_t}\sigma_{t-1}(x) \Rightarrow r_t \leq 2\sqrt{\beta_t}\sigma_{t-1}(x_t)$ where $r_t$ is instantaneous regret at step $t$.*

*Proof.* By definition $r_t = f(x^*) - f(x_t)$ with maxima of $f$ at $x^*$. For $x_t$ to get selected by GP-UCB, it is required that $\mu_{t-1}(x_t) + \sqrt{\beta_t}\sigma_{t-1}(x_t) \geq$

$\mu_{t-1}(x^*) + \sqrt{\beta_t}\sigma_{t-1}(x^*)$ and $\mu_{t-1}(x^*) + \sqrt{\beta_t}\sigma_{t-1}(x^*) \geq f(x^*)$. Then

$$r_t = f(x^*) - f(x_t) \leq \mu_{t-1}(x_t) + \sqrt{\beta_t}\sigma_{t-1}(x_t) - f(x_t) \leq 2\sqrt{\beta_t}\sigma_{t-1}(x)$$

□

**Lemma 2.3.4.** *Define $\boldsymbol{f}_T = [f(x_t)]_{t\in 1\ldots T}$. Information gain of data set at step $T$ can be expressed as*

$$I(\boldsymbol{f}_T; \boldsymbol{y}_T) = \frac{1}{2}\sum_{t=1}^{T} \ln[1 + \sigma^{-2}\sigma_{t-1}^2(x_t)]$$

*Proof.* Following the same paradigm with Formula 2.30 and 2.31, $I(\boldsymbol{f}_T; \boldsymbol{y}_T) = H(\boldsymbol{y}_T) - H(\boldsymbol{y}_T \mid \boldsymbol{f}_T) = H(\boldsymbol{y}_T) - 1/2 \ln|2\pi e\sigma^2 \boldsymbol{I}|$. Also by the chain rule of conditional entropy,

$$I(\boldsymbol{f}_T; \boldsymbol{y}_T) = H(\boldsymbol{y}_{T-1}) + H(y_T \mid \boldsymbol{y}_{T-1}) - 1/2 \ln|2\pi e\sigma^2 \boldsymbol{I}|$$

Knowing $\boldsymbol{y}_{T-1}$ marginalizes the entropy of joint Gaussian $\boldsymbol{y}_T$ into the entropy of $y_T$ alone, so $H(y_T \mid \boldsymbol{y}_{T-1}) = \ln[2\pi e(\sigma^2 + \sigma_{t-1}^2(x_t))]$.

$$\begin{aligned} I(\boldsymbol{f}_T; \boldsymbol{y}_T) &= H(\boldsymbol{y}_{T-1}) + \frac{1}{2}\ln[2\pi e(\sigma^2 + \sigma_{t-1}^2(x_t))] - \frac{1}{2}\ln|2\pi e\sigma^2\boldsymbol{I}| \\ &= H(\boldsymbol{y}_{T-1}) + \frac{1}{2} ln[1 + \sigma_2\sigma_{t-1}^2(x_t)] \\ &\text{By induction towards } T = 1 \text{ and } H(\boldsymbol{y}_0) = 0 \\ &= \frac{1}{2}\sum_{t=1}^{T} ln[1 + \sigma_2\sigma_{t-1}^2(x_t)] \end{aligned} \tag{2.34}$$

□

**Lemma 2.3.5.** *With $\delta$ and $\beta_t$ defined as in Lemma 2.3.2, the following event holds with probability of at least $1 - \delta$.*

$$\sum_{t=1}^{T} r_t^2 \leq \beta_T C_1 I(\boldsymbol{y}_T; \boldsymbol{f}_T) \leq C_1\beta_T\gamma_T \quad \forall T \geq 1, \quad C_1 = \frac{8}{log(1 + \sigma^{-2})}$$

*Proof.* By Lemma 2.3.2 and 2.3.3 the event $\left\{r_t \leq 2\sqrt{\beta_t}\sigma_{t-1}(x_t)\right\} = \left\{r_t^2 \leq 4\beta_t\sigma_{t-1}^2(x_t)\right\}$ holds with probability $\geq 1 - \delta$. Let $s = \sigma^{-1}\sigma_{t-1}(x_t)$ then we have the fact below.

$$\frac{\ln(1 + \sigma^{-2})}{\sigma^{-2}} \leq \frac{\ln(1 + s^2)}{s^2} \text{ for } s \leq \sigma^{-1}$$

This is because $\ln(1+t)/t$ is a decreasing function to $t$. We know $s^2 = \sigma^{-2}k(x_t, x_t) \leq \sigma^{-2}$ since the kernel function is restricted within 1. So we bound $s^2$ by

$$s^2 \leq \frac{\sigma^{-2}\ln(1+s^2)}{\ln(1+\sigma^{-2})}$$

The upper bound of $r_t^2$ then gets written into

$$r_t^2 \leq 4\beta_t\sigma^2\sigma^{-2}\sigma_{t-1}^2(x_t) = 4\beta_t\sigma^2 s^2 \leq 4\beta_t\frac{\ln(1+s^2)}{\ln(1+\sigma^{-2})} \tag{2.35}$$

□

Sum up (2.35) along $t = 1...T$ using Lemma 2.3.4.

$$\begin{aligned}\sum_{t=1}^{T} r_t^2 &\leq \frac{4\beta_T}{\ln(1+\sigma^{-2})}\sum_{t=1}^{T}\ln[1+\sigma^{-1}\sigma_{t-1}(x_t)] \\ &\quad (\beta_t \text{ is non-decreasing}) \\ &= \frac{4\beta_T}{\ln(1+\sigma^{-2})}2I(\boldsymbol{f}_T;\boldsymbol{y}_T) \\ &\leq \frac{8\beta_T}{\ln(1+\sigma^{-2})}\gamma_T = C_1\beta_T\gamma_T\end{aligned} \tag{2.36}$$

By Cauchy–Schwarz inequality,

$$\begin{aligned}R_T^2 = (\sum_{t=1}^{T} r_t)^2 \leq \sum_{t=1}^{T} 1 \sum_{t=1}^{T} r_t^2 = T\sum_{t=1}^{T} r_t^2 \leq TC_1\beta_T\gamma_T \\ \Rightarrow Pr(R_T \leq \sqrt{TC_1\beta_T\gamma_T}) \geq 1-\delta\end{aligned} \tag{2.37}$$

hereby the statement in Theorem 2.3.1 gets proved. Finally $\beta_T = 2\ln\frac{|D|\pi^2t^2}{6\delta}$ needs to be plugged into (2.37) to show the bound of regret sum.

$$TC_1\beta_T\gamma_T = C_1T\gamma_T 2\ln|D| + C_1T\gamma_T 2\ln\frac{\pi^2T^2}{6\delta}$$

Thereby conclusion is that GP-UCB give a sum of regret squares bounded by $O(T\gamma_T\ln|D| + T\gamma_T\ln T) = O^*(T\gamma_T\ln|D|)$ thus completing the proof of Theorem 2.3.1.

According to Srinivas et al., bounds of the maximum information gain $\gamma_T$ vary depending on the choice of kernel function $k$. For the interest of this thesis, we only focus on squared exponential kernel under which $\gamma_T \sim O((lnT)^{d+1})$. This implies strictly sublinear growth of cumulated expected regret by GP-UCB algorithm.

## 2.4 Probabilistic Graphical Models

In this last section of Chapter 2, we make introductory review on principles of probabilistic graphical models, a rich framework of general statistical models based on graph representations. Probabilistic graphical models are powerful toolkits for analyzing ubiquitous uncertainty through structured pattern of reasoning causal and correlation in various machine learning problems. A compelling reason of applying graphical models to probability inference is its compactness of encoding knowledge into proper independence assumption. Consider a highly abstract representation of the world as a joint distribution $\boldsymbol{X} = \{X_i\} = \{X_1, X_2, ..., X_n\}$. With $|X_i|$ as the size of range of a random variable, querying by brute force the arbitrary joint probability $Pr(X_1 = x_1, X_2 = x_2, ..., X_n = x_n)$ requires an estimation of all $\Pi_i|X_i| \sim O(e^n)$ individual parameter combinations and full storage of a size $O(e^n)$ lookup table, which even at small scale makes a majority of problems computationally intractable. Probabilistic graphical models introduce the concept of *factors*, which are essentially non-negative functions $\phi$ with fixed subsets of random variables $\boldsymbol{X}_1, \boldsymbol{X}_2, ..., \boldsymbol{X}_m \subset \boldsymbol{X}$ as inputs. Instead of directly inferring full joint distribution $Pr(\boldsymbol{X}) = Pr(X_1, X_2..., X_n)$, $Pr(\boldsymbol{X})$ are represented in graphical models as a product of factors so that

$$Pr(\boldsymbol{X}) = \frac{\Pi_i \phi_i(\boldsymbol{X}_i)}{Z}, \qquad Z = \sum_{\boldsymbol{x} \in Val(\boldsymbol{X})} \Pi_i \phi_i(\boldsymbol{X}_i) \tag{2.38}$$

with $Z$ as *normalizing constant* or *partition function*. The purpose of $Z$ is to make $Pr(\boldsymbol{X})$ a true probability distribution because $\sum_{\boldsymbol{x} \in Val(\boldsymbol{X})} Pr(\boldsymbol{x}) = 1$. Computing $Pr(\boldsymbol{X})$ is now of complexity $O(e^m)$ and in a properly designed graphical model $m$ is far below $n$. Furthermore, clever inference algorithm can significantly reduce that cost achieving much more efficient ways of querying joint probability.

Another reason that probabilistic graphical models are helpful is their

intuitive declaration of conditional probability. Dependence among random variables are self-explanatory by graph structures and such representation is isolated from the algorithm used to query the posterior probability. The rest of this section introduces two types of graphical models based on directed and undirected graphs, both of which are used in this thesis, followed by review on some common inference algorithms.

### 2.4.1 Directed Models - Bayesian Network

Bayesian networks are effective graph-based representation to compactly encode conditional probability among a large number of dependent random variables. A Bayesian network is a directed acyclic graph $G$ with each of its nodes $\boldsymbol{X}$ standing for a random variable and its edges $\boldsymbol{E}$ specifying directional probability dependence. Every Bayesian network $G(\boldsymbol{X}, \boldsymbol{E})$ satisfies local independence property that for any node $X_i \in \boldsymbol{X}$ given all its parents $Par(X_i)$ it is conditionally independent from its non-descendants.

$$X_i \perp\!\!\!\perp nondescendant_{X_i} \mid Par(X_i) \text{ for } X_i \in \boldsymbol{X} \tag{2.39}$$

#### Reasoning Pattern

The local independence property is derived from *reasoning pattern*, semantics on how probability of nodes influences each other. As previously mentioned the most noticeable advantage of a Bayesian network is its intuitive way of interpreting dependence/independence assumptions among random variables.

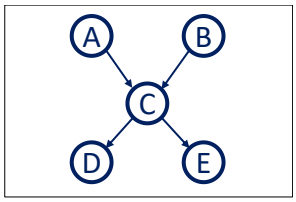



Figure 2.1: Bayesian network example

Figure 2.1 is a simple example of a Bayesian network describing joint probability distribution $Pr(A, B, C, D, E)$. Directed edges are straight forward enough to express the conditional probability $Pr(D|C)$, $Pr(E|C)$, $Pr(C|A)$

and $Pr(C|B)$. When it comes to reasoning pattern, we are more interested in how random variables influence other parts of the graph beyond its immediate neighbors. Figure 2.1 contains 4 important substructures ("trails") in a typical Bayesian network and each substructure represents a unique scenario of reasoning pattern.

- $A \rightarrow C \rightarrow D$ : A influences D if and only if C is not observed. Given no evidence about C, change of probability of A apparently affects C and further flows to D. But an observed C has ascertained value state which blocks influence from A because D in this case is dependent on C only. This pattern is called *causal reasoning*.

- $E \leftarrow C \leftarrow B$ : E influences B if and only if C is not observed. It makes sense to say that event E reflects its immediate cause C which further implies confidence on B. Such backward chains of tracing event causes is called *evidential reasoning*.

- $D \leftarrow C \rightarrow E$ : D influences E if and only if C is not observed. Since both D and E share the common cause, high probability of event D reasonably implies likelihood of event E. However, if C is observed the trail from D to E is blocked as knowing C is sufficient to determining the distribution of E even when event D occurs out of coincidence. Such a pattern is called inter-causal reasoning on *common cause*.

- $A \rightarrow C \leftarrow B$ : This is a funky case in Bayesian network semantics, referred to as "v-structure" in some literature [17]. A influences B if and only if C **is** observed. Counterintuitive as it might seem, no rationale exists for belief on event A to propagate to B simply because they share the same effect C, unless C is known. An intuitive example is when A and B are uniform binary random variables $A, B \in \{0, 1\}$ and C is the xor sum $C = A \cdot \neg B + \neg A \cdot B$. Without presence of precise value of C, $Pr(A = 0 \mid B) = Pr(A = 1) = 0.5$ and $Pr(B = 0 \mid A) = Pr(B = 1) = 0.5$. In other words knowing either A or B does not help determine the state of the other variable. However such independence breaks if $C = 0$ is observed because $A = B$ is necessary condition and likewise $A \neq B$ is necessary for $C = 1$. This "v-structure" semantic is called inter-causal reasoning on *common effect*.

With basic reasoning patterns above, we are able to detect which part of the graph a node may influence in a Bayesian network by traversing those four

different types of "trails". A more formal notation of describing such flow of probability influence is called *d-separation*(direction-dependent separation).

**Definition.** *Let $\boldsymbol{X}_1$, $\boldsymbol{X}_2$ and $\boldsymbol{Z}$ be three non-overlapping subsets of nodes in the same Bayesian network. Then $\boldsymbol{X}_1$, $\boldsymbol{X}_2$ are said to be d-separated by $\boldsymbol{Z}$ if for $\forall X_1 \in \boldsymbol{X}_1$ and $\forall X_2 \in \boldsymbol{X}_2$, $X_1 \perp\!\!\!\perp X_2 \mid \boldsymbol{Z}$.*

In other words, when $\boldsymbol{Z}$ $d$-separates $\boldsymbol{X}_1$ and $\boldsymbol{X}_2$, knowing $\boldsymbol{X}_1$ does not provide additional information about $\boldsymbol{X}_2$ once $\boldsymbol{Z}$ gets observed. The local independence property can be formally restated such that any node in a Bayesian network is $d$-separated from all its non-descendants by its parents, for the obvious reason that any active trail to non-descendants is blocked by its parents.

### Factorization and I-map

The likelihood of a general Bayesian network $G$ consisting of nodes $X_1, ..., X_n$ is computed as $\mathcal{L}(G) = \prod_i^n Pr[X_i|par(X_i)]$. $\mathcal{L}(G)$ is guaranteed to be a legal distribution without need of normalizing constant.

$$\sum_{val(\boldsymbol{X})} \mathcal{L}(G) = \sum_{x_1 \in val(X_1), ..., x_n \in val(X_n)} \prod_i^n Pr[X_i = x_i|par(X_i)] = 1 \qquad (2.40)$$

This is because Bayesian network is a directed acyclic graph so a valid topological sorted order must exist. Without loss of generality, assume $X_1, X_2, ..., X_n$ is topologically sorted. For simplicity $x_1 \in val(X_1)...x_n \in val(X_n)$ is written as $X_1^n$. Then a very simple proof can be constructed as

$$\begin{aligned}
&\sum_{X_1^n} \prod_i^n Pr[X_i = x_i|par(X_i)] \\
&= \sum_{X_1^{n-1}} \Big\{ \prod_i^{n-1} Pr[X_i = x_i|par(X_i)] \sum_{X_n} Pr[X_n = x_n|par(X_n)] \Big\} \\
&= \sum_{X_1^{n-1}} \Big\{ \prod_i^{n-1} Pr[X_i = x_i|par(X_i)] \Big\}
\end{aligned}$$

because $X_n$ is the last node in topological order and it does not serve as a parent for any other nodes. The summation factored out is marginalized on

$val(X_n)$ so it is 1. The conclusion follows by induction. However, $\mathcal{L}(G)$ being legal is not sufficient for a Bayesian network to accurately represent a general joint distribution $Pr(X_1, X_2, ..., X_n)$, unless $Pr(X_1, X_2, ..., X_n)$ *factorizes* the graph $G$.

**Definition.** *A joint distribution $Pr(X_1, X_2, ..., X_n)$ is said to factorize a Bayesian network $G$ if the following condition is satisfied.*

$$Pr(X_1, X_2, ..., X_n) = \mathcal{L}(G) = \prod_i^n Pr[X_i|par(X_i)]$$

In practice we are more interested in knowing under which condition $Pr(X_1, X_2, ..., X_n)$ factorizes $G$ given its Bayesian structure. The answer is that $G$ has to be an *I-map* of $Pr(X_1, X_2, ..., X_n)$.

**Definition.** *Let $I(P)$ be the set of all the conditional independencies that hold in distribution $P \sim Pr(X_1, X_2, ..., X_n)$ and $I(G)$ be the set of all the d-separations in its structure. Then $G$ is said to be an I-map of $P$ if $I(G) \subseteq I(P)$.*

Then we may come to the following conclusion.

**Theorem 2.4.1.** *$G$ is an I-map of distribution $P \Rightarrow P$ factorizes $G$.*

*Proof.* By the chain rule of conditional probability, for general distribution $P \sim Pr(X_1, X_2, ..., X_n)$

$$\begin{aligned} Pr(X_1, X_2, ...X_n) &= Pr(X_n|X_1, ..., X_{n-1})Pr(X_{n-1}|X_1, ..., X_{n-1})...P(X_1) \\ &= Pr(X_1)\prod_{i=2}^n Pr(X_i|X_1, ..., X_{i-1}) \end{aligned}$$

Again w.l.o.g., assume $X_1, ...X_n$ is in topological order so that any parent must have a lower index $i$ than its descendants including children. For any $X_i, i > 1$, the list of nodes $X_1, ...X_{i-1}$ covers all its non-descendants including $par(X_i)$. Moreover, no descendants of $X_i$ could be possibly in $X_1, ...X_{i-1}$. If we annotate $\boldsymbol{Z}_i$ as the non-descendants of $X_i$ excluding $par(X_i)$, the factor $Pr(X_i|X_1, ..., X_{i-1})$ can be written as $Pr[X_i|par(X_i) \cup \boldsymbol{Z}_i]$. Then by property

of local independence in (2.39), $Pr[X_i|par(X_i) \cup \boldsymbol{Z}_i] = Pr[X_i|par(X_i)]$.

$$\begin{aligned} Pr(X_1, X_2, ...X_n) &= Pr(X_1) \prod_{i=2}^{n} Pr[X_i|par(X_i) \cup \boldsymbol{Z}_i] \\ &= Pr(X_1) \prod_{i=2}^{n} Pr[X_i|par(X_i)] = \prod_{i=1}^{n} Pr[X_i|par(X_i)] \end{aligned}$$

As the premise of the theorem states that G is an I-map of P, $Pr[X_i|par(X_i) \cup \boldsymbol{Z}_i] = Pr[X_i|par(X_i)]$ not only holds in the graph but also for the actual distribution $P$. We may conclude that

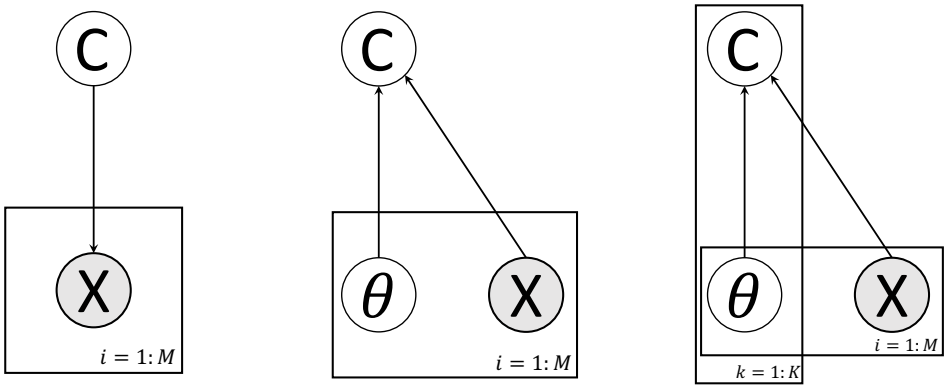
$$P \sim Pr(X_1, X_2, ..., X_n) = \prod_{i=1}^{n} Pr[X_i|par(X_i)] = \mathcal{L}(G)$$

Therefore $G$ factorizes $P$. □

### Plate Notation

In a lot of practical problems rapid growth of complexity of Bayesian network structure makes it difficult to symbolize the graph due to an increasing number of factors and parameters that are necessary for many learning tasks. A great proportion of such complexity comes from repetitive patterns of similar objects that share the same reasoning functionality such as a collection of observations in the same space of joint distribution. Conventionally plate notation is used as a template based language that describes complex Bayesian network structures in compact diagrams, where recursive substructures in the same graph are represented by indexable plates. We discuss two different categories of learning models for classification as motivating examples of using plate nation.

Consider a classification task which requests prediction on objective class $C$ based on observation list $X_1, ..., X_M$. Generative models directly learn a joint distribution $Pr(C, X_1, ..., X_M)$ while discriminative models are only interested in learning $Pr(C|X_1, ..., X_M)$. For binary classification problems arguably the most simplified generative model falls within naive Bayes classifiers which hold strong assumption that observations are mutually independent conditioning on class label $C$. Figure 2.2(a) displays the plate notation of a typical naive Bayes classifier in the case of $M$ observations indexed by




(a) Naive Bayes (Generative)

(b) Logistic Regression (Discriminative)

(c) Max Entropy Classifier (Discriminative)

Figure 2.2: Generative Model vs. Discriminative Model

the same plate. Hence by likelihood of Bayesian networks we verify that

$$\mathcal{L}(G) = Pr(C, X_1, ..., X_M) = Pr(C) \prod_i^M Pr(X_i|C)$$

corresponds to the common semantics of naive Bayes probability. In contrast, for discriminative models, directly sampling $Pr(C|X_1, ..., X_M)$ from observations is difficult because observations are no longer independent given $C$ as indicated by the plate notation in Figure 2.2(b). A parameterized solution helps solve the problem by introducing hidden nodes $\theta$. Note that each $\theta_i$ and $X_i$ are not independent either. So the graph likelihood takes the form of

$$Pr(C|X_1, ..., X_M; \theta) = Pr(C|\boldsymbol{X}; \boldsymbol{\theta})$$

where $\boldsymbol{X}$ and $\boldsymbol{\theta}$ are vectors of $X_i$ and $\theta_i$. A commonly used discriminative approach of materializing $Pr(C|\boldsymbol{X}; \boldsymbol{\theta})$ is the sigmoid function to reformulate the graph likelihood into

$$Pr(C|\boldsymbol{X}; \boldsymbol{\theta}) = \frac{1}{1 + \exp[-\boldsymbol{\theta}^T \boldsymbol{X}]}$$

leading to the familiar logistic regression probability.

The logistic regression model in Figure 2.2(b) can be generalized to multiclass classification by templating $C$ into multiple targets $C_1, ..., C_K$. For each

$C_k$ its conditional distribution is $Pr(C_k|\boldsymbol{X};\boldsymbol{\theta}_k) \propto \exp[-\boldsymbol{\theta}_k^T\boldsymbol{X}]$. Normalizing $Pr(C_k|\boldsymbol{X};\boldsymbol{\theta}_k)$ gives us a maximum entropy (softmax) probability.

$$Pr(C_k|\boldsymbol{X};\boldsymbol{\theta}_k) = \frac{\exp[-\boldsymbol{\theta}_k^T\boldsymbol{X}]}{\sum_{k=1}^{K}\exp[-\boldsymbol{\theta}_k^T\boldsymbol{X}]}$$

The nested plates in Figure 2.2(c) reveal its difference from logistic regression since for every target $C_k$ a set of different parameter list $\theta_k$ is required. Therefore it contains a total of $KM$ parameters.

### 2.4.2 Undirected Models - Markov random field

At the beginning of this section we mentioned the notion of *factors* in the context of graphical models. As factors alone do not necessarily pertain to any conditional probability distribution like Bayesian networks, undirected graphs can be more naturally interpretable models for problems that do not intrinsically bear directionality of probability dependence among random variables. In fact, a joint distribution $P \sim Pr(\boldsymbol{X})$ is called *Gibbs distribution* if $Pr(\boldsymbol{X}) \propto \Pi_i\phi_i(\boldsymbol{X}_i)$, $\boldsymbol{X}_i \subset \boldsymbol{X}$. Markov random fields can be used for directly representing Gibbs distributions in general.

A Markov random field (Markov network) is an undirected graph $G(\boldsymbol{X}, E)$, each of whose edges $(i, j) \in E$ stands for probability dependence between two nodes ($X_i$ and $X_j$). In other words, trails of probability influence may traverse along edges until observed nodes. This leads to *Markov property* as listed below.

- Local Markov property: a node is conditionally independent from all the other nodes given all its neighbors.

$$X_i \perp\!\!\!\perp X_{j\notin adj(i)} \mid \{X_k|k \in adj(i)\}$$

- Pairwise Markov property: given all the other nodes, two nodes must be conditionally independent unless they are neighbors.

$$X_i \perp\!\!\!\perp X_j \mid \{X_k|k \neq i, k \neq j\} \text{ if } (i, j) \notin E$$

- Global Markov property: two non-overlapping node subsets are conditionally independent given a third node subset if nodes in the third

subset blocks all the trails from between the previous two.

$$\boldsymbol{X}_1 \perp\!\!\!\perp \boldsymbol{X}_2 \mid \boldsymbol{X}_3 \text{ where every path between } \boldsymbol{X}_1 \text{ and } \boldsymbol{X}_2 \text{ passes } \boldsymbol{X}_3$$

The global Markov property is generalization of pairwise and local property. Based on simplicity of Markov property, it is reasonable to say that reasoning pattern in a Markov network is typically simpler compared to Bayesian networks due to its undirected nature.

### Factorization and I-map

For a general Markov network $G$ consisting of nodes $X_1, ..., X_n$, the graph likelihood $\mathcal{L}(G)$ can be computed in a product of factors.

$$\mathcal{L}(G) = \frac{\Pi_i \phi_i(\boldsymbol{X}_i)}{Z}, \boldsymbol{X}_i \subset \{X_i\}$$

where $Z$ is the same normalizing constant as defined in (2.38). For every factor $\phi_i(\boldsymbol{X}_i)$, its input $\boldsymbol{X}_i$ has to form a *clique* in the graph. A clique is any fully connnected subgraph of $G$. So factors $\phi_i(\boldsymbol{X}_i)$ are also called *clique potentials* forming $\phi_i(\boldsymbol{X}_i)$ as *clique factorization*. Note that cliques in factorization do not have to be maximal cliques with respect to $G$, meaning that $\boldsymbol{X}_i$ does not have to cover the largest possible fully connected subset of nodes. For the Markov network in Figure 2.3 as an example, both $\mathcal{L}(G) \propto \phi(A, B, D)\phi(A, C)$ and $\mathcal{L}(G) \propto \phi(A, B)\phi(A, D)\phi(B, D)\phi(A, C)$ are valid expressions even though $A, B, D$ is the largest fully connected subcomponent. For a Markov network to hold valid representation of some Gibbs distribution, it is required that the Gibbs distribution factorizes over the graph.

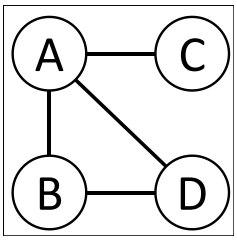



Figure 2.3: Markov network example

**Definition.** *A Gibbs distribution* $P \sim Pr(X_1, X_2, ..., X_n) \propto \Pi_i \phi_i(\boldsymbol{X}_i)$ *is said to factorize the Markov network* $G$ *if* $\forall i, \boldsymbol{X}_i$ *is a clique in* $G$.

This definition is equivalent to saying that the structure of the undirected graph can inspected against every factor in a Gibbs distribution to verify whether a Markov network represents the designated distribution. This is because when a Gibbs distribution $P$ factorizes the $G$, $G$ is an I-map of $P$.

**Definition.** *Let* $I(P)$ *be the set of conditional dependencies that hold in the Gibbs distribution* $P \sim Pr(X_1, X_2, ..., X_n) = \Pi_i \phi_i(\boldsymbol{X}_i)$ *and* $I(G)$ *be the set of all conditional dependencies according to Markov property. Then* $G$ *is said to be an I-map of* $P$ *if* $I(G) \subseteq I(P)$.

The effect is that if Gibbs distribution $P$ factorizes a Markov network $G$, then factors in $P$ satisfy local, pairwise and global Markov property. So we have the theorem as follows.

**Theorem 2.4.2.** *If* $P$ *is a Gibbs distribution factorizing the Markov network* $G$, $G$ *is an I-map of* $P$.

*Proof.* Let $\boldsymbol{X}_1$,$\boldsymbol{X}_2$,$\boldsymbol{Z}$ be any three disjoint subsets of graph nodes $\{X_i\}$ and $\boldsymbol{Z}$ blocks all trails between $\boldsymbol{X}_1$ and $\boldsymbol{X}_2$. We discuss two cases based on $\boldsymbol{X}_1 \cup \boldsymbol{X}_2 \cup \boldsymbol{Z} = \{X_i\}$ and $\boldsymbol{X}_1 \cup \boldsymbol{X}_2 \cup \boldsymbol{Z} \subset \{X_i\}$. First assume $\boldsymbol{X}_1 \cup \boldsymbol{X}_2 \cup \boldsymbol{Z} = \{X_i\}$. Since $P$ factorizes $G$, no factor in $P$ could possibly contain nodes from both $\boldsymbol{X}_1$ and $\boldsymbol{X}_2$ because that violates the rule of cliques on factors. Then $P$ can be written as a $Pr(X_i) \propto \Pi_i \phi_i(\boldsymbol{X}_1) \Pi_j \phi_j(\boldsymbol{X}_2)$ where $\boldsymbol{X}_1 \in \boldsymbol{X_1} \cup \boldsymbol{Z}$, $\boldsymbol{X}_2 \in \boldsymbol{X_2} \cup \boldsymbol{Z}$. Then the factors can be categorized into two larger terms so that $Pr(X_i) \propto \Phi_1(\boldsymbol{X_1}, \boldsymbol{Z}) \Phi_2(\boldsymbol{X_2}, \boldsymbol{Z})$. So $Pr(X_i|\boldsymbol{Z}) \propto \Phi_1(\boldsymbol{X_1}) \Phi_2(\boldsymbol{X_2})$ with $\boldsymbol{Z}$ marginalized out. This proves that $Pr(X_i)$ satisfies any such independence consumption $\boldsymbol{X}_1 \perp\!\!\!\perp \boldsymbol{X}_2 \mid \boldsymbol{Z}$ in $G$.

Next consider the case $\boldsymbol{X}_1 \cup \boldsymbol{X}_2 \cup \boldsymbol{Z} \subset \{X_i\}$. Still no factor possibly contains nodes from both $\boldsymbol{X}_1$ and $\boldsymbol{X}_2$. Let $\boldsymbol{X}_1 \cup \boldsymbol{X}_2 \cup \boldsymbol{Y} \cup \boldsymbol{Z} = \{X_i\}$ and $\boldsymbol{Y}$ does not intersect with the other three sets. This time $P$ can be written as a $Pr(X_i) \propto \Pi_i \phi_i(\boldsymbol{X}_1) \Pi_j \phi_j(\boldsymbol{X}_2)$ where $\boldsymbol{X}_1 \in \boldsymbol{X_1} \cup \boldsymbol{Y} \cup \boldsymbol{Z}$, $\boldsymbol{X}_2 \in \boldsymbol{X_2} \cup \boldsymbol{Y} \cup \boldsymbol{Z}$. Then $Pr(X_i) \propto \Phi_1(\boldsymbol{X_1}, \boldsymbol{Y}, \boldsymbol{Z}) \Phi_2(\boldsymbol{X_2}, \boldsymbol{Y}, \boldsymbol{Z})$. Marginalizing $\boldsymbol{Z}$ gives $Pr(X_i|\boldsymbol{Z}) \propto \Phi_1(\boldsymbol{X_1}, \boldsymbol{Y}) \Phi_2(\boldsymbol{X_2}, \boldsymbol{Y})$. This proves that $Pr(X_i)$ satisfies $\boldsymbol{X}_1, \boldsymbol{Y} \perp\!\!\!\perp \boldsymbol{X}_2, \boldsymbol{Y} \mid \boldsymbol{Z}$. By decomposition of conditional probability [17] this is equivalent to $\boldsymbol{X}_1 \perp\!\!\!\perp \boldsymbol{X}_2 \mid \boldsymbol{Z}$. Thereby the proof is completed. □

Interestingly the inverse of the above theorem is truth as well, i.e., I-map leads to factorization of a Markov network, which is known as the *Hammersley–Clifford theorem*[18].

A particular case of Markov networks is *pairwise Markov network*, which satisfies all Markov property in general except that the likelihood is computed based on connected pairs of nodes. In the case of $G(\boldsymbol{X}, E)$,

$$\mathcal{L}(G) = \Pi_{(i,j)\in E}\phi(X_i, X_j)/Z$$

This expression is attractive in practice as a connected pair of nodes is guaranteed to form a valid clique and no sophisticated graph traversal algorithm is needed due to such simplification. Consider the worst case scenario that a pairwise Markov network of $n$ nodes is fully connected and each node has $m$ discrete possible states. For each of the $n(n-1)/2$ edges there are $m^2$ states so that the parameter space for estimating the distribution is $O(n^2m^2)$. This is far below the size of parameter space $O(m^n)$ required for raw joint distribution of $Pr(X_1, ..., X_n)$ without introducing a graphical model.

**Gaussian Markov Random Field**

Gaussian Markov Random Field (GMRF) is another specialization of Markov networks. A Markov network $G(\boldsymbol{X}, E)$ is a GMRF if the vector $(X_1, X_2, ..., X_n)^T$ is in multivariate normal distribution. The density $Pr(X_i)$ correspond to typical multivariate normal probability.

$$Pr(X_1, X_2, ..., X_n) = (2\pi)^{-n/2}|Q|^{1/2}\exp\{-\frac{1}{2}(\boldsymbol{X}-\boldsymbol{\mu})^T Q(\boldsymbol{X}-\boldsymbol{\mu})\}$$

where $\boldsymbol{\mu}$ are the list of means and $Q$ is the precision matrix computed as the inverse of covariance matrix $\Sigma$ and must satisfy $Q_{ij} > 0$ if and only if $(i,j) \in E$. For single $X_i$ it is normally distributed when marginalized on the rest of the graph. The marginalized distribution of $X_i$ can be estimated as

$$\begin{aligned}
\mathbb{E}(X_i|\{X_{j\neq i}\}) &= \mu_i - \frac{1}{Q_{ii}}\sum_{j:(i,j)\notin E} Q_{ij}(X_j - \mu_j) \\
Prec(X_i|\{X_{i\neq i}\}) &= Q_{ii} \\
Corr(X_i, X_j|\{X_{k\neq i, k\neq j}\}) &= -\frac{Q_{ij}}{Q_{ii}Q_{ij}} = \frac{\Sigma_{ij}}{\sqrt{\Sigma_{ii}\Sigma_{jj}}}, i \neq j \\
Cov(X_i, X_j) &= Q_{ij}^{-1} = \Sigma_{ij}
\end{aligned}$$

Interpreting marginalized distributions from $Q$ is difficult because the covariance term $\Sigma_{ij}$ depends on every term in $Q$. However, constructing $Q$ is straightforward as it can be quickly retrieved by an adjacency table. In case of densely connected graph, $Q$ is mostly positive so computing $Q^{-1}$ is recommended. Moreover, sophisticated sampling approaches are needed for numerically simulating the graph distribution from data because $\mathbb{E}(X_i|\{X_{j\neq i}\})$ does not explicitly specifies how $X_j$ are quantified. Moreover, for graphical models to learn from data we also need concrete inference algorithms for determining posterior distribution.

### 2.4.3 Inference Algorithms

Section 2.4.1 and 2.4.2 make declarative statements on the semantics of several different types probabilistic graphical models, namely how joint probability is represented in the form of a structured set of random variables. But no explanation is explicitly given on how probability is numerically computed through those models. The process of computing numerical probability subject to a well-defined graph is called *inference*, which in the context of graphical models is conventionally formulated as a query task. Given a graphical model standing for $Pr(X_1, ..., X_n)$ we query the graph by asking the question what the exact value $Pr(X_1 = x_1, ..., X_n = x_n)$ is for an arbitrary vector $[x_1, ..., x_n]^T$. Such a query is called *exact inference* because we are interested in knowing the true joint probability of $Pr(X_1, ..., X_n)$. More frequently, we tend to query for $Pr(\boldsymbol{X_i}|\{X\} - \boldsymbol{X}_i), \boldsymbol{X_i} \subset \{X\}$ and in this case the node set $\{X\} - \boldsymbol{X}_i$ is often treated as *evidence* because we are asking for marginal distribution on some nodes used as condition, or $Pr(\boldsymbol{X}|\boldsymbol{E})$ referred to as *conditional probability query*. An even more particular case is about querying the reduced probability $Pr(\boldsymbol{X}|\boldsymbol{E} = \boldsymbol{e})$ where $\boldsymbol{e}$ is known observation of evidence nodes.

There exist two common categories of inference tasks: conditional probability query commonly addressed by variable elimination algorithms and MAP inference often solved as a parameter optimization problem. Variable elimination includes a class of algorithms that compute the target distribution through a sequence of summation or reduction operations on variables. Optimization algorithms directly construct a posterior distribution with minimized distance (e.g. KL divergence) to the target distribution specified by graph semantics. Exact inference in probabilistic graphical models, in worst

cases are always NP-hard problems due to inference complexity that is exponential to scale of the graph [19]. Even approximate inference is of prohibitive complexity because estimating probability of a given state with bounded error – finding $\rho$ subject to $|Pr(\boldsymbol{X} = \boldsymbol{x}|\boldsymbol{E} = \boldsymbol{e}) - \rho| \leq \epsilon$ for some fixed and small $\epsilon$ is NP-hard as well [17]. Fortunately, however, in real-world applications graphical models do not necessarily end with the worst scenario and almost always graph structures can be designed or refactored in a clever way so that inference can be much more efficiently performed. Despite of large literature regarding this topic, this thesis only introduces fundamental ideas how those two categories of inference methods work through a few simple examples and some important procedures that are relevant to subsequent chapters.

### Variable Elimination

We begin discussion of variable elimination algorithm by a simple toy example consisting of nodes $A, B, C, D$, all fully connected. Therefore the corresponding graph supports the following product of factors for joint probability so that no assumption on directionality of the graph is needed.

$$Pr(A, B, C, D) = \frac{1}{Z}\phi(A, B)\phi(A, C)\phi(A, D)\phi(B, C)\phi(B, D)\phi(C, D)$$

For a query task on $Pr(D)$, variable elimination computes $\sum_{a,b,c} Pr(D|a, b, c)Pr(a, b, c)$ as an answer. Starting with $A$, it performs a sequence of what is called *sum-product elimination*.

$$\begin{aligned} P(D) &= \sum_{a,b,c} Pr(D|a, b, c)Pr(a, b, c) \\ &\propto \sum_{a,b,c} \phi(a, b)\phi(a, c)\phi(a, D)\phi(b, c)\phi(b, D)\phi(c, D) \\ &= \sum_{b,c} \phi(b, c)\phi(b, D)\phi(c, D) \sum_{a} \phi(a, b)\phi(a, c)\phi(a, D) \end{aligned} \tag{2.41}$$

If every factor has $d$ assigned states so that summing up $\phi(A, B)\phi(A, C)\phi(A, D)$ on $A$ requires tabular lookup of $O(d^3)$ total entries. Similar elimination goes

for $B$ and $C$.

$$\begin{aligned} P(D) &\propto \sum_{b,c} \phi(b,c)\phi(b,D)\phi(c,D)\phi'(b,c,D) \\ &= \sum_{c} \phi(c,D) \sum_{b} \phi(b,c)\phi(b,D)\phi'(b,c,D) \\ &= \sum_{c} \phi(c,D)\phi''(c,D) = \sum_{c} \phi'''(c,D) \end{aligned} \tag{2.42}$$

The overall complexity $O(d^3)$ is dominated by eliminating $A$ and $B$ as the last two steps only involve two variables. Likewise, a graph of $n$ fully connected nodes has unavoidable inference complexity of $O(d^{n-1})$ where $d$ is typically quadratic to the number of node states. However, most use cases do not compose any fully connected structure and in many cases the graph can be highly sparse if independence assumption is sufficiently and properly asserted. Consider instead of a fully connected graph but a simple linear structure $A - B - C - D$. The reduction chain now turns into an ordered list of arithmetic operations $\sum_a \phi(a,B)$, $\sum_b \phi(b,C)$, $\sum_c \phi(c,D)$ each taking $O(d)$ entry lookups. This results in $O(nd)$ complexity for computing the distribution table of $Pr(D)$, a significantly reduced cost compared to $O(d^{n-1})$ in the fully connected case.

Now it remains to discuss the particular case of reasoning conditional distribution when evidence is available, or the query task of computing $Pr(D|\boldsymbol{E} = \boldsymbol{e})$. For the same example structure of A-B-C-D, a trivial case is that $\boldsymbol{E} = \{A, B, C\}$ and $\boldsymbol{e} = [a, b, c]^T$. As every random variable except $D$ is in the evidence(observation) set, querying $Pr(D|A = a, B = b, C = c)$ does not involve variable elimination and $Pr(D|A = a, B = b, C = c)$ can simply be written as $Pr(D|a, b, c) \propto \phi(a, b)\phi(b, c)\phi(c, D) = \phi'(D)$. A bit more generalized case can be partial evidence so that $\boldsymbol{E} = \{B\}$ and $\boldsymbol{e} = [b]^T$. Expanding the factor expression for $Pr(D|\boldsymbol{E} = \boldsymbol{e})$ gives

$$\begin{aligned} Pr(D|B = b) &\propto \sum_{A,C,D} \phi(A,b)\phi(b,C)\phi(C,D) = \sum_{A,C} \phi'(A)\phi'(C)\phi(C,D) \\ &= \sum_{C} \phi'(C)\phi(C,D) \sum_{A} \phi'(A) \\ &\propto \sum_{C} \phi'(C)\phi(C,D) \propto \phi''(D) \end{aligned}$$

As it turns out, any random variable in the evidence set automatically falls

off the elimination list($\{A, C, D\}$) because they are now constants. Factors that contain evidence can be reduced into a new factor with respect to unobserved variables only. In a more summarized way, this type of sum-product elimination algorithm can be concluded as Algorithm 2 with evidence taken into consideration.

**Algorithm 2** Sum Product Variable Elimination [17]

1: init the factor list $\phi_1, ...\phi_m \in \mathbf{\Phi}$, evidence set $\boldsymbol{E} = \boldsymbol{e}$
2: **if** $\boldsymbol{E} \neq \emptyset$ **then** replace each $\phi_i \in \mathbf{\Phi}$ with $\phi_i(\boldsymbol{E} = \boldsymbol{e})$
3: **end if**
4: set up initial product $\phi(X_1, ...X_n) \leftarrow \prod_i^m \phi_i, \forall X_i, X_i \notin \boldsymbol{E}$
5: **for** $i$ in node 1...n-1 **do**
6: $\quad \phi' \leftarrow$ sum-product-eliminate $\phi_k$ from $\mathbf{\Phi}$ on $X_i$ if $X_i \in scope(\phi_k)$
7: $\quad \mathbf{\Phi} \leftarrow \mathbf{\Phi} \cup \phi'$
8: $\quad \phi(X_{i+1}, ...X_n) \leftarrow \prod_{\psi \in \mathbf{\Phi}} \psi$
9: **end for**
10: return $\phi(X_n)$

The only problem yet remaining to be solved for exact inference using variable elimination algorithm is the partition function $Z$ specified in (2.38). For Bayesian networks, no partition function is required if every such factor $\phi(X \cup par(X))$ is computed according to Bayesian semantics such as $Pr[X|par(X)]Pr[par(x)]$, because their graph likelihood is a legal distribution as proved in (2.40) so $Z = 1$. However, for Markov networks clique factors do not directly constitute conditional probability distributions. Similar techniques as variable elimination can be used for $Z$ with modification, known as "bucket elimination", or *elim-bel* [20] summarized as Algorithm 3.

The main difference between calculating the partition function and conditional probability query is that every variable in the graph is eliminated so that elim-bel algorithm returns a constant instead of a distribution table at completion, in the form of a factor with empty scope $\phi(\emptyset)$, as is desired. Without any optimization, Algorithm 3 runs at the worst complexity of $O(ne^{max_i\{|adj(X_i)|\}})$ where $|adj(X_i)|$ is the number of nodes adjacent to $X_i$ in the graph. This is because the most expensive elimination step occurs at the node with the highest neighbor count so a maximized number of factors get involved in computing the sum of products. In a fully connected graph this complexity is equivalent to $O(ne^n)$. In practice looking for an optimal

**Algorithm 3** Bucket Elimination (elim-bel) [20]

init the factor list $\phi_1, ...\phi_m \in \boldsymbol{\Phi}$, elimination order $X_1, ..., X_n$
set up initial product $\phi(X_1, ...X_n) \leftarrow \prod_i^m \phi_i, \forall X_i, X_i \notin \boldsymbol{E}$
**for** $i$ in node 1...n **do**
    create bucket $\boldsymbol{B}_i \leftarrow \{\phi \mid \phi \in \boldsymbol{\Phi}, X_i \in scope(\phi)\}$
    $\phi'(scope(\boldsymbol{B}_i) \backslash X_i) \leftarrow \sum_{x_i \in X_i} \prod_{\psi \in \boldsymbol{B}_i} \psi$
    $\boldsymbol{\Phi} \leftarrow \boldsymbol{\Phi} \backslash \boldsymbol{B}_i \cup \phi'$
    $\phi(X_{i+1}, ...X_n) \leftarrow \prod_{\psi \in \boldsymbol{\Phi}} \psi$
**end for**
assert $\boldsymbol{\Phi} = \emptyset$
return $Z = \phi(\emptyset)$

elimination order is helpful to find minimized induced graph to save computational cost. Unfortunately finding an exactly optimal order is generally an NP-hard problem as well. A lot of existing research is focused on efficient but greedy algorithms looking for suboptimal solutions such as relying on newly filled edge counts as a cost function or minimized induced width of cliques [21].

### MAP Inference

Unlike conditional probability queries which estimates the objective distribution table through node sequences, maximum a posterior (MAP) inference as a different task attempts to find assignment that maximizes the likelihood of the a graphical model. It addresses probability queries by directly assigning the most probable state to a joint distribution, under the constraint of evidence if any. For a graph of $\boldsymbol{X} = X_1, ..., X_n$, an MAP inference task can simply be defined as finding $\boldsymbol{x}$ subject to

$$\boldsymbol{x} = \underset{\boldsymbol{x}}{\operatorname{argmax}} \, Pr(\boldsymbol{X} = \boldsymbol{x})$$

In case of some evidence set $\boldsymbol{E}$ available, we look for $\boldsymbol{y}$ such that

$$\boldsymbol{y} = \underset{\boldsymbol{y}}{\operatorname{argmax}} \, Pr(\boldsymbol{Y} = \boldsymbol{y} \mid \boldsymbol{E} = \boldsymbol{e}), \boldsymbol{Y} = \boldsymbol{X} \backslash \boldsymbol{E}$$

In terms of factor product,

$$\boldsymbol{x} = \underset{\boldsymbol{x}}{\operatorname{argmax}} \prod_k \phi_k(D_k) = \underset{\boldsymbol{x}}{\operatorname{argmax}} \sum_k \ln \phi_k(D_k), \;\; D_k \text{ is scope}(\phi_k) \tag{2.43}$$

Since looking for globally optimal assignment for MAP is difficult due to large joint probability space, techniques similar to variable elimination can be applied so that factors can sequentially get optimized. The good news is that Formula 2.43 provides us a recipe of maximizing a sum of log factors instead of a product of factors, which makes remarkable contribution to simplifying variable elimination operations. Consider again the toy example of $A - B - C - D$ for $Pr(A, B, C, D) \propto \phi(A, B)\phi(B, C)\phi(C, D)$. For an assignment $[a, b, c, d]^T$ to maximize its likelihood,

$$
\begin{aligned}
& max_{a,b,c,d} \ln[\phi(a, b)\phi(b, c)\phi(c, d)] \\
=& max_{a,b,c,d}[\ln \phi(b, c) + \ln \phi(c, d) + max_a \ln \phi(a, b)] \quad \leftarrow a \text{ is found and eliminated} \\
=& max_{a,b,c,d}[\ln \phi(b, c) + \ln \phi(c, d) + max_a \phi'(b)]
\end{aligned}
$$

The step of finding $a$ is called *max-product elimination* because it maximizes $\phi(a, b)$ subject to $a$ unlike what we did in sum-product elimination which sums up factor outputs based on distinct $a$ values. Following similar steps we are able to discover $b, c, d$ until eventually maximum of $\ln[\phi(A, B)\phi(B, C)\phi(C, D)]$ is computed. In more general cases, this type of max-product elimination techniques can be summarized into the *elim-map* algorithm [20], as listed in Algorithm 4.

**Algorithm 4** Bucket Elimination (elim-map) [20]

1: init the factor list $\phi_1, ...\phi_m \in \boldsymbol{\Phi}$, evidence set $\boldsymbol{E} = \boldsymbol{e}$
2: **if** $\boldsymbol{E} \neq \emptyset$ **then** replace each $\phi_i \in \boldsymbol{\Phi}$ with $\phi_i(\boldsymbol{E} = \boldsymbol{e})$
3: **end if**
4: set up initial product $\phi(X_1, ...X_n) \leftarrow \prod_i^m \phi_i, \forall X_i, X_i \notin \boldsymbol{E}$
5: **for** $i$ in node 1...n **do**
6: create bucket $\boldsymbol{B}_i \leftarrow \{\phi \mid \phi \in \boldsymbol{\Phi}, X_i \in scope(\phi)\}$
7: $\phi'(scope(\boldsymbol{B}_i) \backslash X_i) \leftarrow \max_{x_i \in X_i} \prod_{\psi \in \boldsymbol{B}_i} \psi$
8: $\boldsymbol{\Phi} \leftarrow \boldsymbol{\Phi} \backslash \boldsymbol{B}_i \cup \phi'$
9: add $x_i = \text{argmax}_{x_i} \prod_{\psi \in \boldsymbol{B}_i} \psi$ to results
10: $\phi(X_{i+1}, ...X_n) \leftarrow \prod_{\psi \in \boldsymbol{\Phi}} \psi$
11: **end for**
12: return $[x_1, ..., x_n]^T$

Instead of max-product elimination, this thesis adopts somewhat different strategy of MAP inference from existing approaches for solving the MAB problem, as is explained in detail by subsequent chapters.

# Chapter 3

# Optimizing Bandit Problems with Graphical Models

This chapter starts discussion on the first partition of contribution from this research - a novel predictive model that learns stochastic functions given a limited set of data samples. Interpolation algorithms are commonly seen in supervised learning applications for function approximation by constructing models generalizable to unseen data. However, parametric models such as regression and linear SVMs are limited to functions in the form of predefined algebraic expressions and are thus unsuitable for arbitrary functions without finite number of parameters. Although properly trained neural networks are indeed capable of computing universal functions, the amount of required training data can be prohibitively large in some practical scenarios such as online recommendation. The proposed model addresses both problems based on a semi-parametric graphical model that approximates function outputs with limited data samples through Bayesian optimization. Definition of the graphical model consists of three stages. First, declarative representation of a graphical model is its semantics of how probability gets encoded into the graph structure. Second, to interpret the encoded probability distribution we need to query the graph by inference to compute numerical outputs. Lastly, learning from external data reinforces the model towards better approximation of patterns in data. This chapter covers purely declarative representation and the next two stages are left to subsequent chapters.

## 3.1 Introduction on Contribution

The proposed graphical model addresses the problem of making optimal decisions subject to unknown *environment*, defined as an unknown function without pre-assumed closed-form expression. A typical application where it becomes an imperative optimization task is a recommendation system at cold start. With no prior knowledge as for popularity of its candidate items like online ads, a recommendation engine is expected to quickly grasp an accurate model discriminating between favored and unpopular candidates [22] [23]. Click-through rate (CTR), the ratio of valid user responses and number of viewers (or *impression*) upon an item, is a common metric of item popularity and it can be interpreted as probability that an item achieves user acquisition. Given that CTRs of in-list items are considered as stochastic function outputs, a predictive model can adaptively be applied to learn the unknown environment, whose the location of global optimum is of our best interest. Parametric learning models such as regression do not properly fit this problem because no rigid parameterized assumption is allowed in our problem setting. Neural networks have practically been known to be powerful universal function approximators but as data hungry solutions they require large amount of training data towards predictive functionality. For marketing strategy prediction, acquiring large amount of training data can be prohibitively expensive due to unbounded advertising fees or opportunity cost. It is financially demanded that a recommendation system discover optimal strategy within fewest possible data samples so as to minimize overall marketing cost. This research models this initialization problem as reward optimization with exploration-exploitation trade-off under the paradigm of the multi-armed bandit (MAB) problems [24] [25] [26] in which an agent always seeks for the candidate of the highest CTR. For every decision step, collected reward is sampled and used as feedback to improve future decisions. This online setup is a snapshot of optimal policy search at a single step in typical reinforcement learning scenarios, compared to which MAB considers no state change caused by actions. In addition, budget constraint is imposed on the learning process. In this paper, budget is defined as the total count of recommendation deliveries. Reward is defined as the count of valid user responses (clicks) assuming pay-per-click advertising model.

As one primary component of the two-fold solution contributed by this research, the graphical model that learns from limited CTR samples and

performs inference on stochastic environment through maximum a posteriori (MAP) estimate. The graphical approach refers to candidate items by discrete indices and depicts CTRs of every item into random variables in such a way that item indices are expected to be multivariable to allow for multi-dimensional environment. Rigorous proof is given that the proposed graph complies with fundamental property of a Gaussian Markov random field (GMRF) [27], so that inference can be performed based on multivariate Gaussian covariance between variables. One major advantage of such design is that the model size is constrained by the number of graph nodes as determined by item counts in practice and does not grow with training data, in contrast to traditional Bayesian regression [28] which memorizes all data samples, thus significantly reducing computational complexity particularly when the item count is much smaller than the number of data samples.

The second primary component of our solution is sampling-based decision making policies. As graphical model inference provides the posterior of item rewards, determining the best candidate item break down to a multi-armed bandit problem. The proposed decision making policy is a variation of Thompson sampling [29] [30] that iterates cumulative density across items and selects one that maximizes the expectation that CTR of the sampled item exceeds all the rest. The adapted Thompson sampling method is tested in comparison with traditional meta-approaches on exploration-exploitation trade-off including acquisition functions and the naive epsilon greedy method. Tests reveal that Thompson sampling as decision making policy gives the best outcome in terms of cumulative regret.

The rest of this thesis is structured as follows. In this chapter and the next, the graphical model is defined along with the inference process with proof on its GMRF property and correctness of probability computation. The chapter after the next introduces learning process as sets of decision making policies. In the experiment chapter, the complete framework of graphical model is evaluated against online algorithms in which decision making and model adjustments take place in parallel. The algorithms are experimented on both synthetic environment and real CTR test benches. Using the proposed Bayesian model and online learning algorithm, a real world example is also experimented to showcase effectiveness of our solution on CTR improvement on real data sets. Discussion of experiment results comes right before conclusion.

## 3.2 Model Declaration

### 3.2.1 Problem Nature

Consider the problem of global optimization as follows. Given a set of data samples $\{(i, r_i) \mid r_i = f(i) + \epsilon(i)\}$ where $f$ is a reward function representing environment, we attempt to learn from the reward samples a model $\hat{f}$ mimicking $f$ so that for any $i \in \mathbb{R}^d$, $\hat{f}(i) \approx f(i)$. Sampling noise $\epsilon$ is i.i.d Gaussian noise for all $i \in \mathbb{R}^d$ in this paper. Consequently $r_i$ stands for a random variable subject to some distribution specific to $i$. The sample set in this paper is defined as $\mathbb{S}$.

$$\mathbb{S} = \{(i, r_i) \mid r_i = f(i) + \epsilon(i), i \in D^d\} \tag{3.1}$$

Here $D^d$ is some finite discrete space and $D^d \subset \mathbb{R}^d$. Learning a global approximation of $f(i)$ overkills the key problem in this paper since it only aims at the optimum index $i$ based on $\mathbb{S}$ so that $\text{argmax}_i \hat{f}(i) = \text{argmax}_i f(i)$. Therefore the optimization goal of our interest becomes $i^* = \text{argmax}_{i \in D^d} r_i$. It is important to beware that sampling $(i, r_i)$ from environment incurs extra cost and $\left|\mathbb{S}\right|$ is thereby to be minimized.

### 3.2.2 Graphical Representation

This section describes detailed probability interpretation and inference process of the proposed graphical model, a hybrid graph with similar property from a Markov random field and its subcomponents structured as Bayesian networks.

#### Markov property

Graph construction starts with Markov property among nodes $y_i$ as presented in the example Markov random field by Figure 3.1, where every $y_i$ stands for a *hidden node* or *target node* that infers probability distribution of $r_i$ defined in (3.1). For problems in this paper, we appreciate local Markov property held in a Markov random field. Probability of hidden node $y_v$ is independent from any non-adjacent node given all its neighbors $J_v$.

$$y_v \perp\!\!\!\perp y_{u \notin J_v} \Big| \{y_{u' \in J_v}\}$$

In other words, belief of a hidden node does not propagate beyond adjacent nodes when all its neighbors are certain. The field probability distribution is then computed in clique factorization where a clique is defined as a fully connected subgraph. In this paper cliques are counted based on connected pairwise nodes. The clique joint probability is defined using a Gaussian function $p(y_i, y_{i+1}|\gamma_y) = \exp[-\frac{\gamma_y}{2}(y_{i+1} - y_i)^2]$.

Here $\gamma_y$ constrains the bonding strength between neighbors. Joint density over the field can be a product from all the cliques with $\boldsymbol{y}$ standing for the target node list.

$$\begin{aligned} p(\boldsymbol{y}|\gamma_y) &= \prod_{i=1}^{n-1} \exp(-\frac{\gamma_y}{2}(y_{i+1} - y_i)^2) \\ &= \prod_{i,j \in E} \exp(-\frac{\gamma_y}{2}(y_i - y_j)^2) \end{aligned} \tag{3.2}$$

In general cases, graph nodes $y_i$ are not necessarily linearly connected as in the particular example of Figure 3.1. A more comprehensive expression of Markov joint density is expressed as (3.2) in clique factors from a pairwise Markov network with $E$ being the edge set.

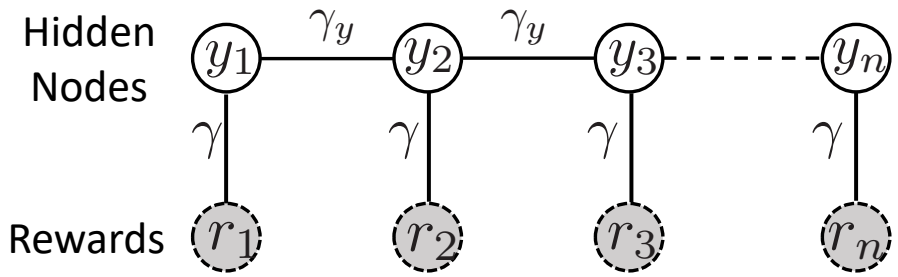



Figure 3.1: Markov property

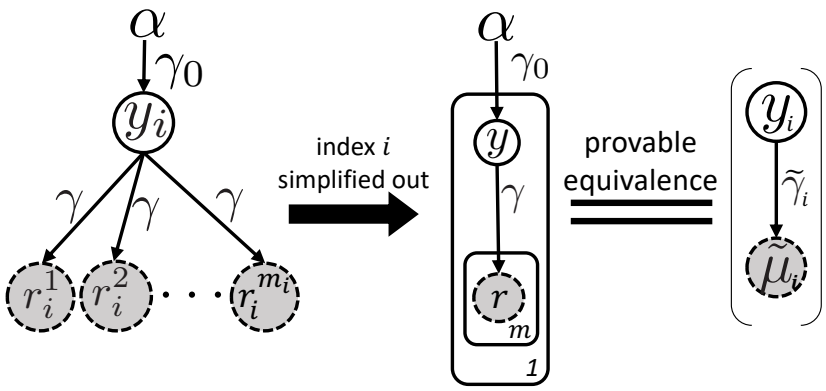



Figure 3.2: Bayesian property

**Bayesian property**

Setup of the Markov network in Figure 3.1 oversimplifies reward patterns by assuming that every hidden node has only one reward sample whose value is certain. In practice, rewards $r_i$ are sampled as a list during learning process and certainty of $y_i$ is under impact from multiple samples. So in addition to joint density over hidden nodes, we introduce Bayesian property by modifying subcomponents under $y_i$ to take care of belief propagation from reward samples to hidden nodes. Figure 3.2 denotes the Bayesian network that expresses such property. Under Bayesian assumption, every target node $y_i$ is conditioned on some prior $\alpha$ with bonding strength $\gamma_0$. The reward list of $y_i$ is represented as nodes $r_i$ separately indexed from 1 to $m_i$. Reward nodes $r_i$ are mutually conditionally independent given their hidden node $y_i$. Similar to the Markov network, bonding coefficient $\gamma$ is assigned between an observed node $r_i$ and its target node $y_i$. Plate notation of the Bayesian network is also provided such that for one target node there are $m$ samples under the parent.

Factorization property of a Bayesian network says the joint distribution of a target node $y_i$ and its children $r_j^{(i)}$ are defined below.

$$p(\boldsymbol{r}^{(i)}, y_i|\gamma, \gamma_0, \alpha) = p(y_i|\gamma_0, \alpha) \prod_{j=1}^{m_i} p(r_j^{(i)}|\gamma, y_i) \tag{3.3}$$

In (3.3), $\boldsymbol{r}^{(i)}$ is the reward list of $y_i$ and $r_j^{(i)}$ is the $j$th sample. Reward list sizes $m_i$ are expected to vary among target nodes. Computing the product

of likelihood over observations using (3.3) gets increasingly expensive as sizes of reward lists grow. Since we are more interested in the distribution of the list $\boldsymbol{r}^{(i)}$ than individual reward samples $r_j^{(i)}$, we approximate every $r_j^{(i)}$ into the mean $\mu_i$ of $\boldsymbol{r}^{(i)}$ so as to eliminate child indices of $y_i$.

$$\begin{aligned} p(\boldsymbol{r}^{(i)}, y_i|\gamma, \gamma_0, \alpha) &= p(y_i|\gamma_0, \alpha)[p(\mu_i|\gamma, y_i)]^{m_i} \\ &= \exp[-\frac{1}{2}\gamma_0(y_i - \alpha)^2] \exp[-\frac{m_i}{2}\gamma(y_i - \mu_i)^2] \\ &\text{where } \mu_i = (\sum_{j=1}^{m_i} r_j^{(i)})/m_i \end{aligned} \tag{3.4}$$

Similar to (3.2), joint density between connected nodes are modeled with Gaussian functions subject to bonding strength $\gamma_0$, $\gamma$ in (3.4). We further work on (3.4) by expanding all the terms in the exponential part.

$$\begin{aligned} &p(\boldsymbol{r}^{(i)}, y_i|\gamma, \gamma_0, \alpha) \\ &= \exp\left\{-\frac{1}{2}[m_i\gamma(y_i - \mu_i)^2 + \gamma_0(y_i - \alpha)^2]\right\} \\ &= \exp\left\{-\frac{1}{2}[(m_i\gamma + \gamma_0)y_i^2 - (2\gamma m_i\mu_i + 2\gamma_0\alpha)y_i + \gamma_0\alpha^2 + m_i\gamma\mu_i^2]\right\} \\ &= \exp\left\{-\frac{1}{2}(m_i\gamma + \gamma_0)[y_i^2 - \frac{2\gamma m_i\mu_i + 2\gamma_0\alpha}{m_i\gamma + \gamma_0}y_i + \frac{\gamma_0\alpha^2 + m_i\gamma\mu_i^2}{\gamma_0 + m_i\gamma}]\right\} \\ &= \exp\left\{-\frac{1}{2}(m_i\gamma + \gamma_0)[y_i^2 - \frac{2\gamma m_i\mu_i + 2\gamma_0\alpha}{m_i\gamma + \gamma_0}y_i + C]\right\} \end{aligned} \tag{3.5}$$

Expression (3.5) indicates that the constant $C$ can easily be scaled so that the exponential term can be rewritten into a perfect square of difference with respect to $y_i$.

$$\begin{aligned} p(\boldsymbol{r}^{(i)}, y_i|\gamma, \gamma_0, \alpha) &= \exp\left\{-\frac{1}{2}\tilde{\gamma}_i(y_i - \tilde{\mu}_i)^2 + C'\right\} \\ &= \exp\{C'\} \exp\left\{-\frac{1}{2}\tilde{\gamma}_i(y_i - \tilde{\mu}_i)^2\right\} \\ &\text{where } \tilde{\gamma}_i = m_i\gamma + \gamma_0 \text{ and } \tilde{\mu}_i = \frac{\gamma m_i\mu_i + \gamma_0\alpha}{m_i\gamma + \gamma_0} \end{aligned} \tag{3.6}$$

Expression (3.6) proves that the complete Bayesian network can be remodeled with only one edge parameter $\tilde{\gamma}_i$ and interpolated sample mean $\tilde{\mu}_i$, whose

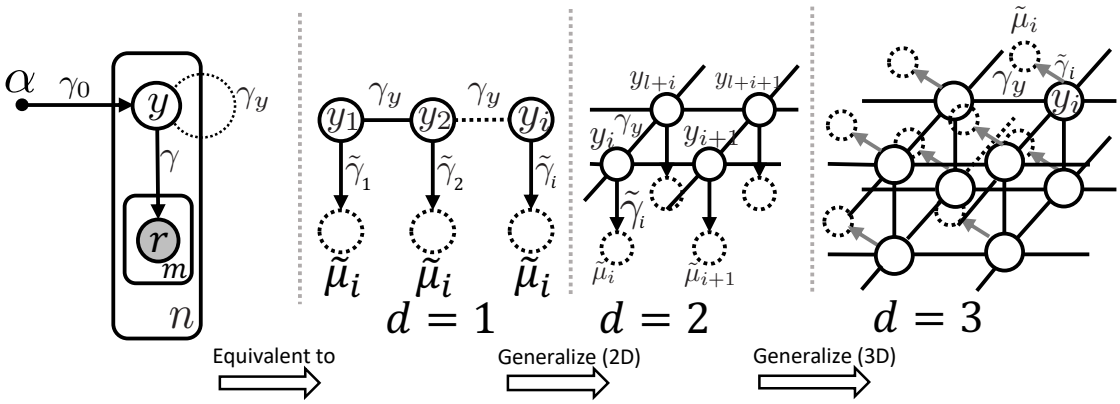



Figure 3.3: Final Graphical Model - Gaussian Markov Random Field

values are $\mu_i$ rescaled by $\gamma$, $\gamma_0$ and $\alpha$. Also notice that $\tilde{\gamma}_i$ differs from constant $\gamma$ since it depends on reward list sizes as given in (3.6). The finally simplified Bayesian network is also displayed in Figure 3.2.

**Final representation**

The final graphical model as Figure 3.3 is a consolidated graph with every hidden node component in Figure 3.1 replaced by the simplified Bayesian representation in Figure 3.2. The plate notation shows an example of $n$ hidden nodes that are bonded with $\gamma_y$, so that the final graphical model is composed of $n$ Bayesian structures in recurrent pattern. This graph is used to infer distinct distributions of every hidden node $y_i$ based on interpolated means $\tilde{\mu}_i$ calculated from samples on $y_i$, meanwhile modeling covariance among $y_i$ as Gaussian kernels. This is equivalent to saying that random vector $\boldsymbol{y} = \langle y_1, y_2, \ldots, y_n \rangle$ is in multivariable Gaussian distribution. Therefore node set $\{y_i\}$ constitutes a Gaussian Markov Random Field (GMRF). Total joint probability $p(\boldsymbol{y}, \boldsymbol{r}|\alpha, \gamma_y, \gamma_0, \gamma)$ of the final graph is computed as product of Bayesian probability in (3.6) on target nodes and Markov joint density counting every edge using (3.2).

$$p(\boldsymbol{y}, \boldsymbol{r}|\alpha, \gamma_y, \gamma_0, \gamma) = p(\boldsymbol{y}|\gamma_y) \prod_i \left[ p(\boldsymbol{r}^{(i)}, y_i|\gamma, \gamma_0, \alpha) \right] \tag{3.7}$$

To make (3.7) explicit, we now plug in both probability factors from (3.6) and (3.2) ignoring the constant factor.

$$p(\boldsymbol{y}, \boldsymbol{r}|\alpha, \gamma_y, \gamma_0, \gamma) \propto \left\{\prod_i \exp\left[-\frac{1}{2}\tilde{\gamma}_i(y_i - \tilde{\mu}_i)^2\right]\right\}\left\{\prod_{i,j\in E} \exp\left[-\frac{\gamma_y}{2}(y_i - y_j)^2\right]\right\} \tag{3.8}$$

In (3.8) $i, j$ refers to an edge connecting $y_i$, $y_j$ given the edge set $E$ and indices $i, j \in D^d$ as indicated in (3.1). A typical GMRF node $y_i$ has an increasing count of neighbors as dimension $d$ goes up as is shown in Figure 3.3. For instance there are 4 neighbors for non-edge nodes in a 2-dimensional graph and 6 in 3-dimensional case. To continue working on (3.8), we take the log likelihood of joint probability $p(\boldsymbol{y}, \boldsymbol{r}|\alpha, \gamma_y, \gamma_0, \gamma)$ as final interpretation of the graph likelihood.

$$\ln p(\boldsymbol{y}, \boldsymbol{r}|\alpha, \gamma_y, \gamma_0, \gamma) \propto \sum_i \left\{-\frac{1}{2}\tilde{\gamma}_i(y_i - \tilde{\mu}_i)^2\right\} + \sum_{i,j\in E} \left\{-\frac{\gamma_y}{2}(y_i - y_j)^2\right\} \tag{3.9}$$

# Chapter 4

# Graphical Inference

## 4.1 Parameter Estimation

This section discusses, on top of the proposed graphical model, how inference is conducted. The parametric model as is declared in Chapter 3 allows for probability inference through parameter estimation. This research adopts optimization based inference, or maximum a posteriori probability (MAP), as one of the two major classes of inference algorithm introduced in Chapter 2 for computing the joint probability distribution. In addition to detailed constructive process of developing a closed-form graph likelihood function, several lemmas are proved for verifying optimality of the solved posterior distribution by the likelihood function.

Now that (3.9) gives a compact representation of model likelihood, inference is now equivalent to maximizing this likelihood with optimal $y_i$ values $\hat{y}_i$, or a target vector $\hat{\boldsymbol{y}}$, subject to currently available reward lists $\boldsymbol{r}$. Hyperparameters $\alpha$, $\gamma_y$, $\gamma_0$, $\gamma$ are initialized with dimension specific values to be stated in experiments. Next we show that a closed-form solution $\hat{\boldsymbol{y}}$ exists for maximizing the model likelihood. Let $E(\boldsymbol{y}) = -2 \ln p(\boldsymbol{y}, \boldsymbol{r} | \alpha, \gamma_y, \gamma_0, \gamma)$ dropping any constant term.

$$E(\boldsymbol{y}) = \sum_i \left\{ \tilde{\gamma}_i (y_i - \tilde{\mu}_i)^2 \right\} + \sum_{i,j \in E} \left\{ \gamma_y (y_i - y_j)^2 \right\} \tag{4.1}$$

The optimal $\hat{\boldsymbol{y}}$ minimizes $E(\boldsymbol{y})$, which can be efficiently computed using

matrix multiplication. Given a graph of $n$ hidden nodes, let $\boldsymbol{B}$ be an $n \times n$ diagonal matrix whose diagonal terms are given by $\tilde{\gamma}_i$ in the list $\tilde{\boldsymbol{\gamma}}$ so that $\boldsymbol{B} = \tilde{\boldsymbol{\gamma}}\boldsymbol{I}_n$. Let $\boldsymbol{K}$ be an $n \times n$ adjacency matrix in which $k_{ij}$ and $k_{ji}$ is $\gamma_y$ if node $y_i$ and $y_j$ are adjacent otherwise 0. In case of high dimensions ($d \geq 2$), graph node indices are flattened before matrix construction so that target nodes are indexable with a 1-dimensional array of size $n$. In this way, $k_{ij} = k_{ji}$ so $\boldsymbol{K}$ is a symmetric matrix whose diagonal terms are set to 0. Furthermore, we define $\boldsymbol{k}$ as a length-$n$ vector consisting of row/column sums of $\boldsymbol{K}$. So $k_x = \sum_j K_{xj} = \sum_i K_{ix}$. Alternatively, $k_x$ can be treated as the neighbor count of the $x$th node. Define matrix $\boldsymbol{A} = \boldsymbol{B} - \boldsymbol{K} + \gamma_y diag(\boldsymbol{k})$ where $diag(\boldsymbol{k})$ is the diagonal matrix with $k_x$ as its $x$th diagonal element.

$$\begin{aligned}\boldsymbol{A} &= \boldsymbol{B} - \boldsymbol{K} + \gamma_y diag(\boldsymbol{k}) \\ &= \begin{pmatrix} \gamma_y k_1 + \tilde{\gamma}_1 & -\gamma_y & \cdots & -\gamma_y & \cdots & \\ -\gamma_y & \gamma_y k_2 + \tilde{\gamma}_2 & \cdots & \cdots & -\gamma_y & \\ \vdots & \vdots & \ddots & & & \\ -\gamma_y & \vdots & & \ddots & & \\ & -\gamma_y & & & \ddots & \\ & & \cdots & & & \gamma_y k_n + \tilde{\gamma}_n \end{pmatrix} \end{aligned} \tag{4.2}$$

$$\boldsymbol{B} = \begin{pmatrix} \tilde{\gamma}_1 & 0 & \cdots & \mathrm{O} \\ 0 & \tilde{\gamma}_2 & & \\ \vdots & & \ddots & \\ \mathrm{O} & & & \tilde{\gamma}_n \end{pmatrix}$$

**Lemma 4.1.1.** *The following scalar produced by the product of nested matrices is equivalent to $E(\boldsymbol{y})$.*

$$(\boldsymbol{y}^\top \tilde{\boldsymbol{\mu}}^\top) \begin{pmatrix} \boldsymbol{A} & -\boldsymbol{B} \\ -\boldsymbol{B} & \boldsymbol{B} \end{pmatrix} \begin{pmatrix} \boldsymbol{y} \\ \tilde{\boldsymbol{\mu}} \end{pmatrix} \tag{4.3}$$

$$= \boldsymbol{y}^\top \boldsymbol{A}\boldsymbol{y} - \tilde{\boldsymbol{\mu}}^\top \boldsymbol{B}\boldsymbol{y} - \boldsymbol{y}^\top \boldsymbol{B}\tilde{\boldsymbol{\mu}} + \tilde{\boldsymbol{\mu}}^\top \boldsymbol{B}\tilde{\boldsymbol{\mu}} \tag{4.4}$$

*Proof.* According to (4.1), $E(\boldsymbol{y}) = \sum_i \{\tilde{\gamma}(y_i - \tilde{\mu}_i)^2\} + \sum_{i,j \in E} \{\gamma_y (y_i - y_j)^2\}$, let $E'(\boldsymbol{y})$ be the declared matrix product.

$$E'(\boldsymbol{y}) \;=\; (\boldsymbol{y}^\top \tilde{\boldsymbol{\mu}}^\top) \begin{pmatrix} \boldsymbol{A} & -\boldsymbol{B} \\ -\boldsymbol{B} & \boldsymbol{B} \end{pmatrix} \begin{pmatrix} \boldsymbol{y} \\ \tilde{\boldsymbol{\mu}} \end{pmatrix}$$

Our task is to prove $E'(\boldsymbol{y})$ is just an alternative expression to $E(\boldsymbol{y})$, or $E(\boldsymbol{y}) = E'(\boldsymbol{y})$ is true.

$$\boldsymbol{B} = \tilde{\boldsymbol{\gamma}} I_n \Rightarrow \tilde{\boldsymbol{\mu}}^\top \boldsymbol{B}\boldsymbol{y} = \boldsymbol{y}^\top \boldsymbol{B}\tilde{\boldsymbol{\mu}} = \sum_{i=1}^{n} \tilde{\gamma}_i y_i \tilde{\mu}_i$$

$$\boldsymbol{B} = \tilde{\boldsymbol{\gamma}} I_n \Rightarrow \boldsymbol{B} = \sum_{i=1}^{n} \tilde{\gamma}_i {\tilde{\mu}_i}^2$$

$\boldsymbol{A}$ is symmetric

$$\Rightarrow \boldsymbol{y}^\top \boldsymbol{A}\boldsymbol{y} = \sum_{i=1}^{n} y_i^2(\gamma_y k_i + \tilde{\gamma}_i) + \sum_{i,j \in E} (-\gamma_y y_i y_j) + \sum_{i,j \in E} (-\gamma_y y_j y_i)$$

$$\Rightarrow \boldsymbol{y}^\top \boldsymbol{A}\boldsymbol{y} = \sum_{i=1}^{n} y_i^2 \gamma_y k_i + \sum_{i=1}^{n} y_i^2 \tilde{\gamma}_i - 2\gamma_y \sum_{i,j \in E} (y_i y_j)$$

$$\Rightarrow E'(\boldsymbol{y}) = \sum_{i=1}^{n} y_i^2 \gamma_y k_i + \sum_{i=1}^{n} y_i^2 \tilde{\gamma}_i - 2\gamma_y \sum_{i,j \in E} (y_i y_j) - 2\sum_{i=1}^{n} \tilde{\gamma}_i y_i \tilde{\mu}_i + \sum_{i=1}^{n} \tilde{\gamma}_i {\tilde{\mu}_i}^2$$

$$\Rightarrow E'(\boldsymbol{y}) = \sum_{i=1}^{n} y_i^2 \gamma_y k_i - 2\gamma_y \sum_{i,j \in E} (y_i y_j) + \sum_{i=1}^{n} \tilde{\gamma}_i (y_i^2 - 2 y_i \tilde{\mu}_i + {\tilde{\mu}_i}^2)$$

$$\Rightarrow E'(\boldsymbol{y}) = \sum_{i=1}^{n} y_i^2 \gamma_y k_i - 2\gamma_y \sum_{i,j \in E} (y_i y_j) + \sum_{i=1}^{n} \tilde{\gamma}_i (y_i - \tilde{\mu}_i)^2$$

$$\Rightarrow E'(\boldsymbol{y}) = \gamma_y (\sum_{i=1}^{n} y_i^2 k_i - 2 \sum_{i,j \in E} (y_i y_j)) + \sum_{i=1}^{n} \tilde{\gamma}_i (y_i - \tilde{\mu}_i)^2 \tag{4.5}$$

Recall that the setup of matrix $\boldsymbol{A}$ requires that $k_x$ be the neighbor count of the target node at index $x$. $\sum_{i=1}^{n} y_i^2 k_i$ is literally for every node its value squared times its neighbor count which is the number of edges it connects. From the edge perspective every edge $(y_i, y_j)$ is counted twice respectively

by $y_i$ and $y_j$. So we have

$$\begin{aligned}
&\sum_{i=1}^{n} y_i^2 k_i = \sum_{i,j\in E} y_i^2 + \sum_{i,j\in E} y_j^2 \\
&\Rightarrow E'(\boldsymbol{y}) = \gamma_y(\sum_{i,j\in E} y_i^2 - 2\sum_{i,j\in E}(y_i y_j) + \sum_{i,j\in E} y_j^2) + \sum_{i=1}^{n} \tilde{\gamma_i}(y_i - \tilde{\mu_i})^2 \\
&\Rightarrow E'(\boldsymbol{y}) = \sum_{i,j\in E} \gamma_y(y_i - y_j)^2 + \sum_{i=1}^{n} \tilde{\gamma_i}(y_i - \tilde{\mu_i})^2 \\
&\Rightarrow E'(\boldsymbol{y}) = \sum_{i} \tilde{\gamma_i}(y_i - \tilde{\mu_i})^2 + \sum_{i,j\in E} \gamma_y(y_i - y_j)^2
\end{aligned} \tag{4.6}$$

Therefore $E(\boldsymbol{y}) = E'(\boldsymbol{y})$ is true. □

With Lemma 4.1.1, we are able to compute the negative log likelihood $E(\boldsymbol{y})$ directly from matrix $\boldsymbol{A}$ and $\boldsymbol{B}$. Recall that $\boldsymbol{y}$ is the target node list and $\tilde{\boldsymbol{\mu}}$ is the interpolated sample means of target nodes. $\left(\boldsymbol{y}^\top \tilde{\boldsymbol{\mu}}^\top\right)$ stands for horizontal concatenation of vectors $\boldsymbol{y}^\top$ and $\tilde{\boldsymbol{\mu}}^\top$ and similarly $\boldsymbol{y}$ and $\tilde{\boldsymbol{\mu}}$ can be vertically concatenated as well. Hence Formula (4.3) produces a product of three matrices of sizes $1 \times 2n$, $2n \times 2n$ and $2n \times 1$. In order to maximize the graph likelihood, we need to find the optimal value $\hat{\boldsymbol{y}}$ that minimizes $E(\boldsymbol{y})$ and equivalently maximizes the model likelihood. Solving $\hat{\boldsymbol{y}}$ depends on the following lemma.

**Lemma 4.1.2.** *$E_{min} = E(\hat{\boldsymbol{y}}) = \tilde{\boldsymbol{\mu}}^\top(\boldsymbol{B} - \boldsymbol{B}\boldsymbol{A}^{-1}\boldsymbol{B})\tilde{\boldsymbol{\mu}}$ and $\hat{\boldsymbol{y}} = \boldsymbol{A}^{-1}\boldsymbol{B}\tilde{\boldsymbol{\mu}}$.*

*Proof.* For this proof, we assume $\boldsymbol{A}$ is positive definite, which as another lemma will get proved later. A positive definite $\boldsymbol{A}$ has a unique Cholesky decomposition $\boldsymbol{D}^\top\boldsymbol{D}$ and $\boldsymbol{A} = \boldsymbol{D}^\top\boldsymbol{D}$. Also define $\boldsymbol{z} = \boldsymbol{D}\boldsymbol{y}$. The objective is to prove that a closed-form solution exists to minimize $E(\boldsymbol{y})$ below.

$$E(\boldsymbol{y}) = \boldsymbol{z}^\top\boldsymbol{z} - \tilde{\boldsymbol{\mu}}^\top\boldsymbol{B}\boldsymbol{D}^{-1}\boldsymbol{z} - \boldsymbol{z}^\top{\boldsymbol{D}^{-1}}^\top\boldsymbol{B}\tilde{\boldsymbol{\mu}} + \tilde{\boldsymbol{\mu}}^\top\boldsymbol{B}\tilde{\boldsymbol{\mu}}$$

subject to

$$E_{min} = \tilde{\boldsymbol{\mu}}^\top(\boldsymbol{B} - \boldsymbol{B}\boldsymbol{A}^{-1}\boldsymbol{B})\tilde{\boldsymbol{\mu}}$$

As $\boldsymbol{B}$ is effectively a diagonal matrix with diagonal elements from $\tilde{\boldsymbol{\gamma}}$ it is symmetric as well. With regard to the expression above, $E(\boldsymbol{y})$ can be written as factorization plus some remainder irrelevant to $\boldsymbol{z}$.

$$
\begin{aligned}
E(\boldsymbol{y}) &= \boldsymbol{z}^\top(\boldsymbol{z} - \boldsymbol{D}^{-1^\top}\boldsymbol{B}\tilde{\boldsymbol{\mu}}) - \tilde{\boldsymbol{\mu}}^\top\boldsymbol{B}\boldsymbol{D}^{-1}\boldsymbol{z} + \tilde{\boldsymbol{\mu}}^\top\boldsymbol{B}\tilde{\boldsymbol{\mu}} \\
&= \boldsymbol{z}^\top(\boldsymbol{z} - \boldsymbol{D}^{-1^\top}\boldsymbol{B}\tilde{\boldsymbol{\mu}}) - \tilde{\boldsymbol{\mu}}^\top\boldsymbol{B}\boldsymbol{D}(\boldsymbol{z} - \boldsymbol{D}^{-1^\top}\boldsymbol{B}\tilde{\boldsymbol{\mu}}) - \tilde{\boldsymbol{\mu}}^\top\boldsymbol{B}\boldsymbol{D}^{-1}\boldsymbol{D}^{-1^\top}\boldsymbol{B}\tilde{\boldsymbol{\mu}} + \tilde{\boldsymbol{\mu}}^\top\boldsymbol{B}\tilde{\boldsymbol{\mu}} \\
&= (\boldsymbol{z}^\top - \tilde{\boldsymbol{\mu}}^\top\boldsymbol{B}\boldsymbol{D}^{-1})(\boldsymbol{z} - \boldsymbol{D}^{-1^\top}\boldsymbol{B}\tilde{\boldsymbol{\mu}}) + \tilde{\boldsymbol{\mu}}^\top\boldsymbol{B}\tilde{\boldsymbol{\mu}} - \tilde{\boldsymbol{\mu}}^\top\boldsymbol{B}\boldsymbol{D}^{-1}\boldsymbol{D}^{-1^\top}\boldsymbol{B}\tilde{\boldsymbol{\mu}} \\
&= (\boldsymbol{y}^\top\boldsymbol{D}^\top - \tilde{\boldsymbol{\mu}}^\top\boldsymbol{B}\boldsymbol{D}^{-1})(\boldsymbol{D}\boldsymbol{y} - \boldsymbol{D}^{-1^\top}\boldsymbol{B}\tilde{\boldsymbol{\mu}}) + \tilde{\boldsymbol{\mu}}^\top(\boldsymbol{B} - \boldsymbol{B}\boldsymbol{A}^{-1}\boldsymbol{B})\tilde{\boldsymbol{\mu}} \\
&= (\boldsymbol{y}^\top - \tilde{\boldsymbol{\mu}}^\top\boldsymbol{B}\boldsymbol{D}^{-1}\boldsymbol{D}^{\top^{-1}})\boldsymbol{D}^\top\boldsymbol{D}(\boldsymbol{y} - \boldsymbol{D}^{-1}\boldsymbol{D}^{-1^\top}\boldsymbol{B}\tilde{\boldsymbol{\mu}}) + \tilde{\boldsymbol{\mu}}^\top(\boldsymbol{B} - \boldsymbol{B}\boldsymbol{A}^{-1}\boldsymbol{B})\tilde{\boldsymbol{\mu}} \\
&= (\boldsymbol{y}^\top - \tilde{\boldsymbol{\mu}}^\top\boldsymbol{B}\boldsymbol{A}^{-1})\boldsymbol{D}^\top\boldsymbol{D}(\boldsymbol{y} - \boldsymbol{A}^{-1}\boldsymbol{B}\tilde{\boldsymbol{\mu}}) + \tilde{\boldsymbol{\mu}}^\top(\boldsymbol{B} - \boldsymbol{B}\boldsymbol{A}^{-1}\boldsymbol{B})\tilde{\boldsymbol{\mu}}
\end{aligned}
\tag{4.7}
$$

In the above expression the transpose matrix of $\tilde{\boldsymbol{\mu}}^\top\boldsymbol{B}\boldsymbol{A}^{-1}$ is $\boldsymbol{A}^{-1^\top}\boldsymbol{B}^\top\tilde{\boldsymbol{\mu}}$. As $\boldsymbol{A}$ is symmetric $\boldsymbol{A}^{-1}$ is symmetric as well. Therefore $\boldsymbol{A}^{-1^\top}\boldsymbol{B}^\top\tilde{\boldsymbol{\mu}} = \boldsymbol{A}^{-1}\boldsymbol{B}\tilde{\boldsymbol{\mu}}$.

Now it is easy for us to have $[(\boldsymbol{y}^\top - \tilde{\boldsymbol{\mu}}^\top\boldsymbol{B}\boldsymbol{A}^{-1})\boldsymbol{D}^\top]^\top = \boldsymbol{D}(\boldsymbol{y}^\top - \tilde{\boldsymbol{\mu}}^\top\boldsymbol{B}\boldsymbol{A}^{-1})^\top = \boldsymbol{D}(\boldsymbol{y} - \boldsymbol{A}^{-1}\boldsymbol{B}\tilde{\boldsymbol{\mu}})$, which indicates a dot product of the same vector, or the $L2$-norm of vector $\boldsymbol{D}(\boldsymbol{y} - \boldsymbol{A}^{-1}\boldsymbol{B}\tilde{\boldsymbol{\mu}})$.

$$
E(\boldsymbol{y}) \;=\; [\boldsymbol{D}(\boldsymbol{y} - \boldsymbol{A}^{-1}\boldsymbol{B}\tilde{\boldsymbol{\mu}})]^2 + \tilde{\boldsymbol{\mu}}^\top(\boldsymbol{B} - \boldsymbol{B}\boldsymbol{A}^{-1}\boldsymbol{B})\tilde{\boldsymbol{\mu}}
$$

Minimizing $E(\boldsymbol{y})$ requires that $(\boldsymbol{y} - \boldsymbol{A}^{-1}\boldsymbol{B}\tilde{\boldsymbol{\mu}}) = 0$. Now we are ready for $\hat{\boldsymbol{y}} = \boldsymbol{A}^{-1}\boldsymbol{B}\tilde{\boldsymbol{\mu}}$ and $E_{min} = \tilde{\boldsymbol{\mu}}^\top(\boldsymbol{B} - \boldsymbol{B}\boldsymbol{A}^{-1}\boldsymbol{B})\tilde{\boldsymbol{\mu}}$ thus completing proof of lemma 4.1.2. □

lemma 4.1.2 assumes positive definiteness on $\boldsymbol{A}$, a necessary condition for $\boldsymbol{A}$ to possess unique Cholesky decomposition in the form of $\boldsymbol{D}^\top\boldsymbol{D}$. This assumption is proved below.

**Lemma 4.1.3.** *$\boldsymbol{A}$ is positive definite.*

**Definition.** *A symmetric $n \times n$ matrix $\boldsymbol{A}$ is positive definite if the scalar $\boldsymbol{v}^\top\boldsymbol{A}\boldsymbol{v}$ is strictly positive for arbitrary non-zero vector $v$.*

$$
\boldsymbol{v}^\top\boldsymbol{A}\boldsymbol{v} > 0 \text{ where } \boldsymbol{v} \in \mathbb{R}^n \text{ and } \exists v_i \neq 0
$$

*Proof.* Let $v$ be a size-$n$ non-zero real vector. Based on how $\boldsymbol{A}$ is constructed

in (4.2), $\boldsymbol{v}^\top \boldsymbol{A}\boldsymbol{v}$ can be expanded as follows.

$$\begin{aligned}\boldsymbol{v}^\top \boldsymbol{A}\boldsymbol{v} &= \sum_{i=1}^{n} v_i^2(\gamma_y k_i + \tilde{\gamma}_i) + \sum_{i,j\in E}(-\gamma_y v_i v_j) + \sum_{i,j\in E}(-\gamma_y v_j v_i) \\ &= \sum_{i=1}^{n} v_i^2\gamma_y k_i + \sum_{i=1}^{n} v_i^2\tilde{\gamma}_i - 2\sum_{i,j\in E}\gamma_y v_i v_j \ (E \text{ is the edge set})\end{aligned} \tag{4.8}$$

As defined by (3.6) $\tilde{\gamma}_i = m_i\gamma + \gamma_0$. $m_i$ is the sample size of target node $y_i$. As hyperparameters $\gamma$ and $\gamma_0$ are always set to be positive and $\exists v_i \neq 0$, we prove that $\sum_{i=1}^{n} v_i^2\tilde{\gamma}_i > 0$. Applying the similar trick used in the proof of Lemma 4.1.1, we know $\sum_{i=1}^{n} v_i^2\gamma_y k_i$ iterates all target nodes in the graph and sums up $v_i^2\gamma_y$ times neighbor count of node $y_i$. Again from an edge perspective, every edge $i, j \in E$ is counted twice so that $\sum_{i=1}^{n} v_i^2\gamma_y k_i = \sum_{i,j\in E} v_i^2\gamma_y + \sum_{i,j\in E} v_j^2\gamma_y$.

$$\begin{aligned}&\sum_{i=1}^{n} v_i^2\gamma_y k_i - 2\sum_{i,j\in E}\gamma_y v_i v_j = \sum_{i,j\in E} v_i^2\gamma_y + \sum_{i,j\in E} v_j^2\gamma_y - 2\sum_{i,j\in E}\gamma_y v_i v_j \\ &\Rightarrow \sum_{i=1}^{n} v_i^2\gamma_y k_i - 2\sum_{i,j\in E}\gamma_y v_i v_j = \sum_{i,j\in E}\gamma_y(v_i - v_j)^2 \\ &\Rightarrow \boldsymbol{v}^\top \boldsymbol{A}\boldsymbol{v} > \sum_{i=1}^{n} v_i^2\gamma_y k_i - 2\sum_{i,j\in E}\gamma_y v_i v_j = \sum_{i,j\in E}\gamma_y(v_i - v_j)^2 \geq 0\end{aligned} \tag{4.9}$$

Therefore $\boldsymbol{v}^\top \boldsymbol{A}\boldsymbol{v} > 0$ is true. $\boldsymbol{A}$ is positive definite. □

With lemma 4.1.1, 4.1.2 and 4.1.3 we now strictly prove that $\hat{\boldsymbol{y}}$ is the optimal parameter values to define the posterior distribution by the desginated graphical reresentation.

$$\hat{\boldsymbol{y}} = \underset{\boldsymbol{y}}{\text{argmax}}\left[\log p(\boldsymbol{r}, \boldsymbol{y}|\gamma_y, \gamma, \gamma_0, \alpha)\right] \tag{4.10}$$

$$= \boldsymbol{A}^{-1}\boldsymbol{B}\tilde{\boldsymbol{\mu}} \text{ and } \boldsymbol{\sigma}_{\hat{\boldsymbol{y}}}^2 = diag(\boldsymbol{A}^{-1}) \tag{4.11}$$

Here $\boldsymbol{\sigma}_{\hat{\boldsymbol{y}}}^2$ is the posterior variance of $\hat{\boldsymbol{y}}$ taken from diagonal terms of $\boldsymbol{A}^{-1}$. Given $\boldsymbol{y} = \boldsymbol{A}^{-1}\boldsymbol{B}\tilde{\boldsymbol{\mu}}$, $E(\boldsymbol{y}) = E_{min} = \tilde{\boldsymbol{\mu}}^\top(\boldsymbol{B} - \boldsymbol{B}\boldsymbol{A}^{-1}\boldsymbol{B})\tilde{\boldsymbol{\mu}}$. Let $\Lambda = \boldsymbol{B} -$

$\boldsymbol{B}\boldsymbol{A}^{-1}\boldsymbol{B}$. The model probability distribution is expected to be

$$p(\boldsymbol{r}, \boldsymbol{y}|\gamma, \gamma_y, \gamma_0, \alpha) \propto \exp\left(-\frac{1}{2}\tilde{\boldsymbol{\mu}}^\top \Lambda \tilde{\boldsymbol{\mu}}\right) \tag{4.12}$$

Formula (4.12) proves that the final graph in Figure 3.3 conforms to GMRF property with multivariate Gaussian distribution such that $p(\boldsymbol{y}) \sim \mathcal{N}(\tilde{\boldsymbol{\mu}}, \Lambda^{-1})$. $\Lambda$ serves as the precision matrix.

Lastly, computing $\boldsymbol{A}$ defined in (4.2) requires sums of rows in $\boldsymbol{K}$. This leads to context overhead every time the model gets updated. We found that approximating $\boldsymbol{A}$ with $\boldsymbol{A'}$ greatly improves computational efficiency in practice.

$$\begin{aligned}
\boldsymbol{A'} &= \boldsymbol{B} - \boldsymbol{K} + 2d\gamma_y \boldsymbol{I}_n \\
&= \begin{pmatrix}
2d\gamma_y + \tilde{\gamma_1} & -\gamma_y & \cdots & -\gamma_y & \cdots & & \\
-\gamma_y & 2d\gamma_y + \tilde{\gamma_2} & \cdots & \cdots & -\gamma_y & & \\
\vdots & \vdots & \ddots & & & & \\
-\gamma_y & \vdots & & \ddots & & & \\
 & -\gamma_y & & & \ddots & & \\
 & & \cdots & & & 2d\gamma_y + \tilde{\gamma_n}
\end{pmatrix}
\end{aligned} \tag{4.13}$$

$A'$ replaces neighbor counts $k_x$ on the diagonal with constant $2d$. This in effect avoids counting neighbors by assuming that every $d$-dimensional GMRF node has $2d$ neighbors, an assumption true for all except edge nodes. Therefore we call approximation with $A'$ an *edge normalization* method because edge nodes are treated as if they were non-edge nodes.

To summarize, the following closed-form solution is adopted as model inference for later experiments in this thesis.

$$\hat{\boldsymbol{y}} = \underset{\boldsymbol{y}}{\operatorname{argmax}} \left(\log p(\boldsymbol{r}, \boldsymbol{y}|\gamma_y, \gamma, \gamma_0, \alpha)\right) = \boldsymbol{A'}^{-1}\boldsymbol{B}\tilde{\boldsymbol{\mu}} \tag{4.14}$$

$$\boldsymbol{\sigma}^2_{\hat{\boldsymbol{y}}} = diag(\boldsymbol{A'}^{-1}) \tag{4.15}$$

$\hat{\boldsymbol{y}}$ is model prediction with uncertainty measured by the variance list $\boldsymbol{\sigma}^2_{\hat{\boldsymbol{y}}}$.

## 4.2 Relevance to the Bandit Problem

The multi-armed bandit (MAB) problem as the core problem of this thesis has been systematically introduced in Chapter 2. The nature of MAB is widely studied and applied to many decision making problems [31] [32] [33] associated with environment reward. This section only briefly reviews the problem description and how the graphical model designed in this research can be of benefit for solving the MAB problem from Bayesian perspective.

Given a list of discrete random variables called "arms" $\{Y_i\}$ whose values represent sample rewards. In general, $\{Y_i\}$ are not of i.i.d, which leads to the problem nature how to select the arm of maximum expected reward. Given $T$ rounds of attempts, an ideal policy collects maximized reward from the best arm. The loss due to failure in collecting optimal reward is measured in regret. MAB is formally defined as follows.

Given a random variable list $\{Y_i\}$ and $\mu_i$ as mean of $Y_i$, a policy decides the index $\pi(t)$ of the chosen arm at the $t$th step and $r_i^{(t)}$ is the sample reward observed on the $i$th arm at step $t$. Under policy $\pi(t)$, the total regret after $T$ rounds of observation is defined as $R_T = T\mu^* - \sum_{t=1}^{T} r_{\pi(t)}^{(t)}$ and $\mu^* = \max_k \mu_k$. True values of mean $\mu_i$ in MAB are never accessible so $\mu^*$ can only be approximated through inference. This paper models $Y_i$ as $r_i$ defined in (3.1). We assume $\{Y_i\} \sim \mathcal{N}(\tilde{\boldsymbol{\mu}}, \Lambda^{-1})$ which the proposed graph conforms to. With graph node $\boldsymbol{y_i}$ corresponding to belief in $r_i$, the goal of optimization mentioned in Section 3.2.1 can be redefined under the bandit setting as minimizing the regret sum $\sum_{t=1}^{T} r_{i^*} - r_{\pi(t)}$ over $T$ iterations.

# Chapter 5

# Graphical Learning

Regulating learning tasks for a probabilistic graphical model is the last stage of defining an intact predictive model after declarative representation and inference mentioned in Chapter 3. It is of prioritized concern to clearly state our goal of learning tasks first before we decide upon choice of learning methods which may vary depending on learning tasks. For the key point of this thesis, we have to focus on tuning the graphical model that best possibly fit the criteria of solution to the MAB problem.

There are two primary approaches of learning graphical models. The first attempts to acquire a model with optimal network structure and is thus referred to as "structural learning" in some literature. Constructing a proper graphical model practically requires training on a large scale of data to learn a distribution of candidate network structures from which the model that is the most likely to be optimal is selected. In most use cases it consumes high loads of expert knowledge even to build a candidate model collection of moderate size. Therefore in this research we restrict our graphical model to a fixed structure given a well defined problem and every node complies with one-to-one mapping to a candidate arm. In this way we exclude structural learning from trainable consideration so as to avoid violation of budget constraint governed by bandit problems.

The second learning approach learns model parameters assuming a fixed structure. This learning approach is often associated with two different prediction tasks. The first prediction task is expected to learn from a training set $\boldsymbol{r}$ to approximate a joint distribution $P(\boldsymbol{y}, \boldsymbol{r})$, where $\boldsymbol{y}$ are the trainable model parameters corresponding to the distributions of candidate arms.

This learning process is known as "generative training" because the joint distribution is directly taken as the prediction objective, which is typically achieved by generative models mentioned in preceding discussion in Section 2.4.1. The second prediction task involves "discriminative training", which enables the model to grasp a close approximation of $Pr(\boldsymbol{y}|\boldsymbol{r})$. Typical discriminative models such as those discussed in Section 2.4.1 often separately encode all the necessary conditional probability distributions represented by a Bayesian network. For the proposed graph representation (Figure 3.3) we adopt a variation to generative training by taking all the Bayesian substructures into consideration and working on maximizing the joint probability $Pr(\boldsymbol{y}, \boldsymbol{r})$ in (4.14).

Apart from clarifying the goal of learning task, we further need to consider additional functional requirements particular to the MAB problem. First, to minimize the size of training set we favor online learning due to its adaptiveness mentioned in Section 2.1. It allows for stepwise model adjustment as reward sampling goes on, so that graph inference updates itself based on (4.14) whenever new information about the reward list $\boldsymbol{r}$ is available. This calls for the need in decision making policy that decides the index of node/arm to sample from at every step. Second, to come up with a uniformly good (refer to Section 2.2.1) policy the exploration-exploitation trade-off has to be weighed with caution to ensure a problem independently effective solution. In this chapter, we introduce several different versions of decision making policy used for learning and optimizing the graphical model.

## 5.1 Decision Making Policy

Decision making policy serves as rules necessary to regulate which action to take at every step, hopefully in order to achieve optimal sequential actions. Concretely, given model inference $\hat{\boldsymbol{y}}^{(t)}$, $\boldsymbol{\sigma}_{\hat{\boldsymbol{y}}}^{(t)}$ by Formula (4.14) (4.15) at time $t$, policy is needed to produce $\pi^{(t)}$, a decision index at which the next sampling takes place. It now resorts to solving an $n$-armed bandit problem at every step $t$ in the sense that there are $n$ target nodes in the graph of Figure 3.3 to choose from. A decision index $\pi^{(t)}$ either exploits current model inference by trusting the largest node in $\hat{\boldsymbol{y}}^{(t)}$ or put more emphasis on exploring further rewards from nodes with higher uncertainty based on variance list $\boldsymbol{\sigma}_{\hat{\boldsymbol{y}}}^{(t)}$. This research covers three suites (meta-policy) of decision making policy as listed

below. Later experiments are designed to verify which policy achieves the most satisfactory regret evaluation.

### 5.1.1 Acquisition Function

Constructing an acquisition function (ACQ) is a common approach in Bayesian optimization to determine the next optimal index to sample at. An acquisition function is usually an inexpensive utility function that commensurates posterior distributions and measures the desirability of each distribution. Commonly used acquisition functions include probability of improvement [34] (PI), expected improvement [35][36] (EI) and upper confidence bound [37] (UCB). Let $a(\hat{\boldsymbol{y}}, \boldsymbol{\sigma}_{\hat{\boldsymbol{y}}})$ be the generic stereotype of an acquisition function that performs element-wise operation on input vectors and returns a vector of the same size so that policy produces $\pi \leftarrow \text{argmax}_i\, a(\hat{\boldsymbol{y}}, \boldsymbol{\sigma}_{\hat{\boldsymbol{y}}})$. Below is parameter setting for the three acquisition functions used in this paper.

- probability of improvement

$$a_{PI}(\hat{\boldsymbol{y}}, \boldsymbol{\sigma}_{\hat{\boldsymbol{y}}}) = \Phi(\frac{\hat{\boldsymbol{y}} - r_{best} - \xi}{\boldsymbol{\sigma}_{\hat{\boldsymbol{y}}}})$$

where $r_{best}$ is current best (largest) sample reward across all the nodes so far. $\Phi$ is the standard Gaussian cumulative density. The quantity of probability of improvement $a_{PI}(\hat{y}, \sigma_{\hat{y}})$ is essentially the probability that $\hat{y}$ is greater than the best of all the samples currently observed because it is easy to show that $\Phi[(\hat{y} - r_{best} - \xi)/\sigma_{\hat{y}}] = Pr(\hat{y} > r_{best} + \xi)$. Restricting the constant bias $\xi$ to a small value can be of minor impact on the confidence of desirability. In this research, we persist in $\xi = 0.01$.

- expected improvement

$$a_{EI}(\hat{\boldsymbol{y}}, \boldsymbol{\sigma}_{\hat{\boldsymbol{y}}}) = z\boldsymbol{\sigma}_{\hat{\boldsymbol{y}}}\Phi(z) + \boldsymbol{\sigma}_{\hat{\boldsymbol{y}}}\phi(z)$$

where $z = (\hat{\boldsymbol{y}} - r_{best} - \xi)/\boldsymbol{\sigma}_{\hat{\boldsymbol{y}}}$ and $\phi$ is the standard Gaussian probability density. For $\sigma_{\hat{y}} = 0$, $a_{EI}(\hat{y}, \sigma_{\hat{y}}) = 0$. Much less straightforward as its expression appears, EI shares similar semantics with PI and is also based on the metrics of how likely a given sample surpluses currently the best record. With $\sigma_{\hat{y}}\phi(z)$ adding to the acquisition measure, nodes that are more frequently

sampled tend to have lower variance thus yielding next sampling opportunity to uncertain nodes and securing the exploration side of the trade-off balance.

- Gaussian process - upper confidence bound

$$a_{GP-UCB}(\hat{\boldsymbol{y}}, \boldsymbol{\sigma}_{\hat{\boldsymbol{y}}}) = \hat{\boldsymbol{y}} + \sqrt{\beta(x)}\boldsymbol{\sigma}_{\hat{\boldsymbol{y}}}$$

where $x$ is total number of current observations and $\beta(x) = 2log(dx^2\pi^2/6\delta)$ [12]. $d$ is dimension of node indices and $\delta = 0.9$. This acquisition function is exactly the same as the one used by the GP-UCB algorithm proposed by Srinivas et al. thoroughly introduced in Section 2.3.2 and 2.3.3. Although GP-UCB is considered a common option for Gaussian process regression, this research tentatively takes it for comparative experiments to verify how well it applies to GMRF based models since this work shared the same problem nature with works from Srinivas et al.

### 5.1.2 Epsilon Greedy

The same naive classical approach introduced in Section 2.2.3, Epsilon greedy (EPS) is a classical and naive exploration-exploitation trade-off heuristic that allocates a probability $1 - \epsilon$ for exploitation behavior and $\epsilon$ for else. In this research $\epsilon = 0.2$.

$$\pi \leftarrow \begin{cases} \text{argmax}_i\, \hat{\boldsymbol{y}}, & \text{with probability } 1 - \epsilon \\ \text{random } i \in D^d, & \text{with probability } \epsilon \end{cases} \tag{5.1}$$

### 5.1.3 Thompson Sampling

The general form of Thompson Sampling (TS) as a classical heuristic has been introduced in Section 2.2.4. For the particular case of our graphical inference, it is as simple as generating random numbers based on the estimated model parameters. As a result, Thompson Sampling tends to trust posterior distribution of parameters and from every node $y_i$ takes a random sample based on $\mathcal{N}(\hat{y}_i, \sigma_i)$. Greedy as the algorithm might seem, such pure sampling is very effective to less explored arms due to larger standard deviation.

$$\pi \leftarrow \underset{i}{\text{argmax}} \{y_i | y_i \sim \mathcal{N}(\hat{y}_i, \sigma_i)\}$$

In this research we also design a more sophisticated variation of Thompson sampling dedicated to further adapt the policy for application in practical CTR optimization. Details are discussed along with experiment setup in the next chapter.

# Chapter 6

# Experiments and Evaluation

Incorporating full functionality of the completely defined GMRF based graphical model, we are ready to deploy this model into solving general multi-armed bandit problems. To testify how well this model works in terms of expected cumulative regret it achieves, three suites of experiments are conducted for separate evaluation. The first batch of experiments are purely based on the original MAB problem setting, for which some theoretical "black-box" function are synthesized as hidden ground truth environment we expect the model to predict. For tests on synthesized functions, environment of 1-$d$, 2-$d$ and 3-$d$ is separately experimented to illustrate how resilient our predictive model is against higher-dimensional sparse data samples. Evaluations on these tests compare between meta-policies defined in Chapter 5 to show how fast each policy reaches convergence of cumulative regret.

The second batch of experiments takes advantage of experiment design on synthesized functions and generalizes the bandit setting to CTR optimization for online advertising. Concretely, using the same predictive model and similar experiment setup, an ad recommendation algorithm is designed specifically for traffic scheduling so that the candidate ad with the highest CTR may receive the largest share of recommendation (impression). Successful traffic rescheduling is expected to significantly improve the total number of user clicks given that CTRs of candidate ads are correctly predicted. A testbench that simulates real-world clicking behavior is used in these experiments and CTR logs generated by the testbench serve as training samples. Similar to experiments on synthesized environment, the recommendation algorithm

is tested on environment of 1-$d$, 2-$d$ and 3-$d$ supported by the testbench. Still, comparative evaluations between different policies are conducted after modification to these policies.

The third batch of experiments turn to real-world dataset which is collected from click logs from a large number of content distributors. The purpose of real data experiments is to showcase the possibility of practical installation of all the works involved in this research despite existing limitation that needs to be addressed.

Lastly, we make more detailed elaboration on the modified version of Thompson sampling used in ad recommendation algorithm concerning two aspects. First, unoptimized full-spectrum sampling requires repetitive calculation of integrals on a large number of transcendental functions, which suffers from prohibitively high computational cost, because the major modification we made to the original sampling algorithm (Section 5.1.3) is to turn the index based procedure into a listwise sampling approach. We thereby analyze the complexity of the modified Thompson sampling and propose more realistic numeric approximation for satisfying similar analytic purpose. The second point involves simple tricks of improving numerical stability on top of illustration on numeric approximation.

## 6.1 Experiment Setup

### 6.1.1 Hyperparameter Tuning

The four hyperparameters $\alpha$, $\gamma_y$, $\gamma_0$, $\gamma$ are manually tuned through preliminary tests based on synthesized functions separate from those used in our experiments to ensure effective learning capability of the graphical model. As $\alpha$ is no more than some prior conventionally installed in Bayesian networks, we found during tuning process that setting $\alpha = 0$ does not significantly impact model inference but we preferred some tiny value of $\alpha$. Based on Formula (3.6) it is clear that $\gamma_0$ and $\gamma$ serve as smoothing parameters for interpolated means $\tilde{\mu}_i$ so we want $\gamma_0$ to be much smaller than $\gamma$ to avoid over-scaling $\mu_i$. Since magnitude of $\gamma$ is responsible of connection strength $\tilde{\gamma}_i$ we experimented on $\gamma$ with several different orders including 0.01, 0.1, 1.0 and found that $\gamma = 0.01$ achieves satisfactory inference in reconstructing unknown functions in one-dimensional cases. Furthermore $\tilde{\mu}_i$ essentially

specifies the precision of the Gaussian function in (3.6) so it controls how much confidence hidden nodes gain from observations. Increasing $\gamma$ makes the Gaussian function skinnier with respect to $y_i$ so that the graph is less confident on observations because likelihood drops faster as $y_i$ deviates from the $\tilde{\mu}_i$. This is typically desired when the environment dimension goes up because we intend the model to request slightly more observations to deal with sparsity of the index space caused by higher dimensions. Therefore we decide $\gamma = 0.02d$, $\gamma_0 = 0.01\gamma$ to be dimension dependent hyperparameters where $d$ is the dimension count.

We also found that $\gamma_y$ is the most sensitive hyperparameter to the proposed graph. Intuitively it affects correlation among target nodes and pretty much resembles the scale parameter in radial-basis kernel function in Gaussian process. We opted for $\gamma_y = 0.01$ in this paper as we found that large $\gamma_y$ cripples inter-node belief propagation, which is to be avoided when adjacent nodes are expected to be correlated arms; oppositely too small $\gamma_y$ grants too much correlation to the Markov random field and this makes the graphical model inclined to make incorrect inference on the truly optimal arm whose neighbors are of low rewards. In extreme cases, huge $\gamma_y$ completely prevents a target node from influencing neighborhood and tiny $\gamma_y$ allows confidence to flow freely due to too much correlation.

### 6.1.2 Experiment 1 - Learning Synthetic Functions

Wrapping up the learning model and meta-policies, we present our first complete online algorithm as Algorithm 5 in which unknown environment is learnt through incremental reward sampling. For each iteration, the policy decides upon graphical inference the location to sample at, so that the sampled reward as a training datum helps update model inference which in turn improves decision making in upcoming iterations. The similar process goes on for a few dozens of iterations before the model parameters come close enough for approximating the environment by predicting the optimal index to achieve regret convergence.

The four hyperparameters mentioned in Section 3.2 require initialization before any learning takes place. Throughout this thesis unless otherwise stated, $\alpha = 0.001$, $\gamma_y = 0.01$, $\gamma = 0.02d$, $\gamma_0 = 0.01\gamma$, $d$ representing index dimension, i.e., dimension of environment as previously stated. Algorithm 5 is repeated on three different dimensions of environment so that $d = 1, 2, 3$.

**Algorithm 5** Synthetic Function Learning

1: Initialize hyperparameters $\gamma$, $\gamma_y$, $\gamma_0$, $\alpha$
2: Random $\hat{\boldsymbol{y}}$, $\boldsymbol{\sigma}_{\hat{\boldsymbol{y}}} \leftarrow \mathcal{N}(0, \sigma^2)$
3: Average regret $\bar{R}_t \leftarrow 0$ ▷ evaluation only
4: **for** $t$ in iteration 1...T **do**
5: $\pi^{(t)} \leftarrow$ from policy ▷ $ACQ/EPS/TS$
6: Sample reward at index $\pi^{(t)}$ as $r_{\pi^{(t)}}(t) \leftarrow f(\pi^{(t)})$
7: Update graph likelihood $\hat{\boldsymbol{y}}$, $\boldsymbol{\sigma}_{\hat{\boldsymbol{y}}} \leftarrow p(\mathbf{r}, \mathbf{y}|\gamma, \gamma_y, \gamma_0, \alpha)$
8: Current regret $R_t \leftarrow max_i f(i) - r_{\pi^{(t)}}(t)$ ▷ evaluation only
9: Update $\bar{R}_t \leftarrow \bar{R}_t \frac{t-1}{t} + R_t \frac{1}{t}$ ▷ evaluation only
10: **end for**

For each case, node index $i \in D^d$. Prior distributions of all nodes are initialized as standard Gaussian. During every iteration, the policy is responsible for the current index of sampling the next reward and this policy comes from one of the three meta-policies in Section 5.1. For evaluation purpose only, Algorithm 5 also keeps track of regret in every iteration and computes the cumulative average from iteration 1 to $t$ as $\bar{R}_t$ (Line 3, 8, 9). Ideally average regret converges to 0 after enough rounds of iterations passed. Experiment 1 is practically designed to compare different performance among policies in terms of minimum average regret and how fast this final average regret is achieved. The synthesized function $f$ used as environment varies depending on $d$ and settings are respectively given below.

- 1D Environment

$$f(i) = \phi(i; \mu_1, \sigma_1^2) + \phi(i; \mu_2, \sigma_2^2) + \phi(i; \mu_3, \sigma_3^2) + \epsilon$$
$$\mu_1 = -3, \mu_2 = -1, \mu_3 = 3$$
$$\sigma_1^2 = 0.15, \sigma_2^2 = 1.5, \sigma_3^2 = 0.7$$

- 2D Environment

$$f(i) = \phi(i; \boldsymbol{\mu}_1, \Sigma_1) + \phi(i; \boldsymbol{\mu}_2, \Sigma_2) + \phi(i; \boldsymbol{\mu}_3, \Sigma_3) + \epsilon$$

$$\boldsymbol{\mu}_1 = \begin{pmatrix} -3 \\ 3 \end{pmatrix}, \boldsymbol{\mu}_2 = \begin{pmatrix} 0 \\ 0 \end{pmatrix}, \boldsymbol{\mu}_3 = \begin{pmatrix} 3 \\ -3 \end{pmatrix}$$

$$\Sigma_1 = \begin{pmatrix} 0.025 & 0 \\ 0 & 0.025 \end{pmatrix}, \Sigma_2 = \begin{pmatrix} 2.25 & 0 \\ 0 & 2.25 \end{pmatrix}, \Sigma_3 = \begin{pmatrix} 0.25 & 0 \\ 0 & 0.25 \end{pmatrix}$$

- 3D Environment

$$f(i) = 3.0[\phi(i; \boldsymbol{\mu}_1, \Sigma_1) + \phi(i; \boldsymbol{\mu}_2, \Sigma_2) + \phi(i; \boldsymbol{\mu}_3, \Sigma_3)] + \epsilon$$

$$\boldsymbol{\mu}_1 = \begin{pmatrix} 3 \\ 3 \\ 3 \end{pmatrix}, \boldsymbol{\mu}_2 = \begin{pmatrix} -2 \\ -2 \\ -2 \end{pmatrix}, \boldsymbol{\mu}_3 = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$$

$$\Sigma_1 = \begin{pmatrix} 3 & 0 & 0 \\ 0 & 3 & 0 \\ 0 & 0 & 3 \end{pmatrix}, \Sigma_2 = \begin{pmatrix} 0.5 & 0 & 0 \\ 0 & 0.5 & 0 \\ 0 & 0 & 0.5 \end{pmatrix}, \Sigma_3 = \begin{pmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 2 \end{pmatrix}$$

Environment $f(i)$ is the sum of three Gaussian density functions with independent means and variances. Sampling noise $\epsilon \sim \mathcal{N}(0, 0.025)$ and is applied to all types of environment used in experiments to introduce randomness to $f(i)$. The above setting of $f(i)$ leads to three maxima and the tested algorithm is challenged to discover the global maxima $\mu_1$, $\boldsymbol{\mu}_1$ and $\boldsymbol{\mu}_2$ in 1D, 2D and 3D environment, where multivariate Gaussian density is administered in composition of higher dimensions.

For environment of each dimension, index space $D^d$ is selected as discrete linear grid on the interval $X = [-5.0, 5.0]$ so that $|D^d|$ is close to 1000. For example when $d = 1$ the grid increment is 0.01. For $d = 2, 3$ node indices are formed by $X \times X$ and $X \times X \times X$ with larger grid increments. As for iteration numbers $T$ as budget, Experiments for 1D and 2D environment has $T = 1150$ and $T = 1075$ iterations for 3D.

### 6.1.3 Experiment 2 - Learning Recommendation Scheduling

Experiments on synthesized environment shows how a theoretical function can be learnt from scratch through reward sampling. Nevertheless, such experiment design faces considerable limitation when it comes to predicting click rates. In practical use cases, decisions are not necessarily sequentially performed as in original MAB problem setting, such that content distribution occurs in parallel among multiple ads and collecting only one CTR sample at every iteration is a huge waste of potential clicks. For operational content

**Algorithm 6** Learning Recommendation Scheduling

1: Initialize hyperparameters $\gamma$, $\gamma_y$, $\gamma_0$, $\alpha$
2: Random $\hat{\boldsymbol{y}}$, $\boldsymbol{\sigma}_{\hat{\boldsymbol{y}}} \leftarrow \mathcal{N}(0, \sigma^2)$
3: Impression weights $w_i^{(0)} \leftarrow 1/n$
4: Average regret $\bar{R}_t \leftarrow 0$ ▷ evaluation only
5: **for** $t$ in iteration 1...T **do**
6: $\{w_i^{(t)}\} \leftarrow$ from modified policy ▷ $ACQ/EPS/TS$
7: Collect #clicks $\{C_i^{(t)} | C_i^{(t)} = \chi(\widetilde{f}(i), Nw_i^{(t)})\}$ from test bench $\chi$ ▷ online
8: $\{r_i^{(t)}\} \leftarrow$ calculate CTRs $C_i^{(t)}/Nw_i^{(t)}$
9: **for** $i$ in index 1...n **do** ▷ offline
10: Sample reward at node index $i$ as $r_i^{(t)}$
11: Update $\hat{\boldsymbol{y}}$, $\boldsymbol{\sigma}_{\hat{\boldsymbol{y}}} \leftarrow p(\mathbf{r}, \mathbf{y} | \gamma, \gamma_y, \gamma_0, \alpha)$
12: **end for**
13: Current regret $R_t \leftarrow Nmax_i \widetilde{f}(i) - \sum_i C_i^{(t)}$ ▷ evaluation only
14: Update $\bar{R}_t \leftarrow \bar{R}_t \frac{t-1}{t} + R_t \frac{1}{t}$ ▷ evaluation only
15: **end for**

distribution algorithm thereof, an efficient way of scheduling recommendation traffic is to collect CTR samples from every candidate ad and have the predictive model analyze the statistics offline. Instead of a single decision index, we prefer to develop some policy that produces a weight list standing for the proportions of total traffic to get assigned to candidate ads. Knowing the CTR posteriors of all the candidate items, the model predicts a weight list to redirect larger share of traffic to ads with high predictions. This scenario can concretely be described as finding optimal solution of assigning $w_i N$ impressions for every item $i$ to collect maximized user clicks. Algorithm 6 is presented in testbench experiments as a modified version of Algorithm 5. Regarding the experiments, $N = 100,000$ in analogy to 100,000 ads as available budget. Algorithm 6 differs from Algorithm 5 mainly in such a way that it isolates sampling, an online activity from inference, an offline activity as noted in Line 9. We expect Algorithm 6 to assign $w_i N$ ad deliveries on item $y_i$ given $n$ candidate items represented by $n$ graph nodes. In every iteration, the offline inference does not incur any advertising cost once online sampling is done.

The way a weight list $w_i$ gets constructed depends on which type of meta-policy ($ACQ/EPS/TS$) is used as is defined in each case below.

**Weights from Acquisition Function**

$$\big\{w_i | w_\pi = 1.0, w_{i\neq\pi} = 0\big\}, \pi = \underset{i}{\operatorname{argmax}}\, a(\hat{\boldsymbol{y}}, \boldsymbol{\sigma}_{\hat{\boldsymbol{y}}})$$

$w_i$ is essentially a one-hot vector set at optimal index given by the acquisition function.

**Weights from Epsilon Greedy**

$$\big\{w_i | w_\pi = 1 - \epsilon,\ w_{i\neq\pi} = \epsilon/(n-1)\big\}, \pi = \underset{i}{\operatorname{argmax}}\{y_i | y_i \in \hat{\boldsymbol{y}}\}$$

The above weight assignment is a direct interpretation of the probability that a node is selected when a single decision index is to be made under classical $\epsilon$-greedy policy. It is fairly easy to see that $\sum_i w_i = 1.0$.

**Weights from Modified Thompson Sampling**

As posterior distributions $\hat{\boldsymbol{y}}, \boldsymbol{\sigma}_{\hat{\boldsymbol{y}}}$ in fact deliver more information than plain prediction on means values $\boldsymbol{y}$, from which Thompson sampling can be improved through maximized expectation estimation. Suppose a reward sample from node $y_i = \hat{y}_i$. The probability of node $i$ being the largest in the graph $p(\hat{y}_i >= y_{j\neq i}) = \prod_{j\neq i} p(\hat{y}_i >= y_j)$. Note that covariance among nodes is not considered at sampling time. Let $\Phi_j$ be Gaussian cumulative density of node $y_j$ so that $\Phi_j(\hat{y}_i) = p(\hat{y}_i >= y_j)$. Then $p(\hat{y}_i >= y_{j\neq i})$ is computed as below.

$$p(\hat{y}_i >= y_{j\neq i}) = \prod_{j\neq i} \Phi_j(\hat{y}_i) \tag{6.1}$$

Formula (6.1) is an approximation of de facto reward distributions which are not independent among nodes. With such approximation, we are able to compute expectation of $p(\hat{y}_i >= y_{j\neq i})$ as follows.

$$\begin{aligned} E[p(y_i >= y_{j\neq i})] &= \int_{-\infty}^{\infty} p(\hat{y}_i >= y_{j\neq i})\ d\hat{y}_i & (6.2)\\ &= \int_{-\infty}^{\infty} \prod_{j\neq i} \Phi_j(\hat{y}_i)\ d\hat{y}_i & (6.3) \end{aligned}$$

Therefore weights $w_i$ are a proportion list by expectation in (6.2).

$$\{w_i / \sum_k w_k | w_i = E[p(y_i >= y_{j \neq i})]\}$$

The above weights assign impression traffic based on expected probability of each node being optimal, such that for a single click the assigned node is multinomially distributed as is specified by those weights.

The CTR test bench $\chi$ used in Experiment 2 simulates real life Web advertising environment by generating random user clicks in the following way. For candidate item $y_i$, given its assigned impression $Nw_i$ and click probability $\widetilde{f}(i)$ (Line 7), a recommendation attempt either receives valid user response or not, as a Bernoulli trial. Consequently the number of clicks $C_i$ on item $y_i$ is in binomial distribution of $B(Nw_i, \widetilde{f}(i))$, where $\widetilde{f}$ is the 1-0 max-min scaled version of function $f$ over its discrete domain. Environment $f$ used for the test bench is the same to those defined in Section 6.1.2. Hyperparameters are initialized with the same values in Algorithm 5. Interval increment $X$ used in Experiment 2 is adjusted so that $|D^d| \approx 100$ instead of 1,000 as in former experiments. This raises environment sparsity to make it harder for the model to pick accurate CTR estimation.

## 6.2 Evaluation

### 6.2.1 Synthetic Data Evaluation

Figure 6.1, 6.2 and 6.3 demonstrate cumulative regrets from experiments in all three cases of distinct $d$. For each case, 30 repeated trials are conducted and averaged for statistically significant evaluation. It is revealed that Thompson sampling wins final $\bar{R}_T$ for all three cases. Some policies fail to locate the global maximum by getting stuck at local minima (as caused by larger variances), such as acquisition function PI for all three cases and EI for $d = 3$.

In case of policy failure, exploration simply ceases before global maximum is located in testing environment. In such cases, the policy leads to linear total regret as is expressed cumulative average regret staying constant.

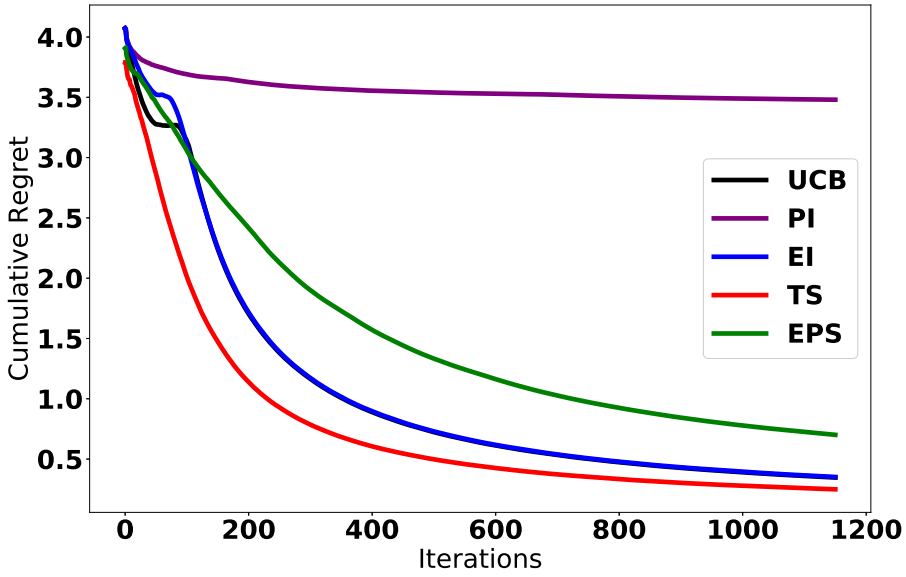



Figure 6.1: 1D Test - Synthesized Environment

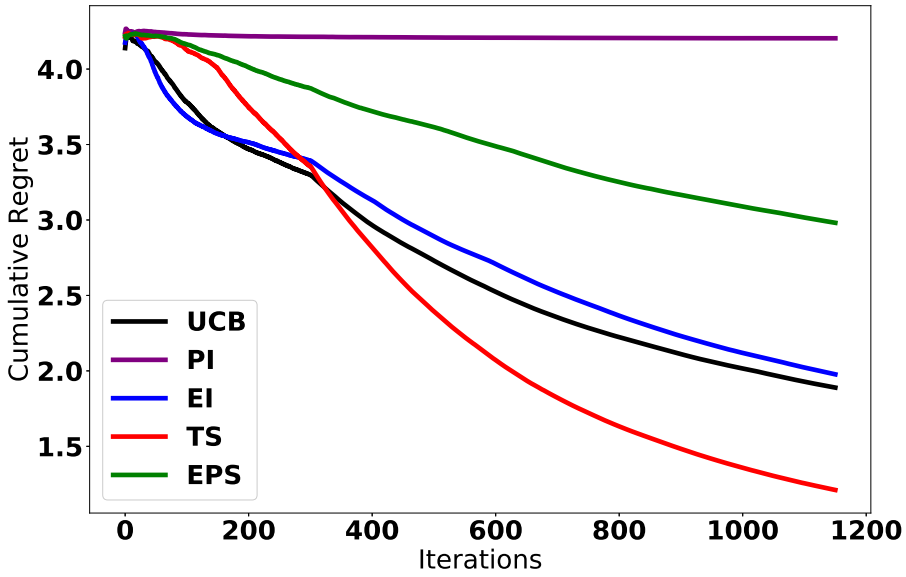



Figure 6.2: 2D Test - Synthesized Environment

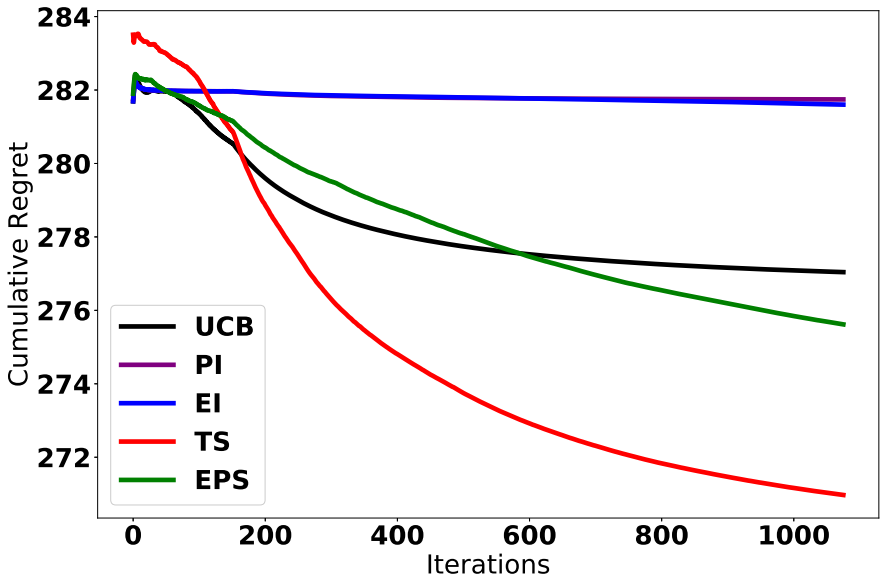



Figure 6.3: 3D Test - Synthesized Environment

### 6.2.2 Testbench Data Evaluation

Similar to Experiment 1, environment in different dimensions are respectively tested and evaluation is averaged across 30 trials. Figure 6.4, 6.5 and 6.6 display average missing clicks as regrets under different policies. Since Algorithm 6 performs sampling across all graph nodes, it draws much faster regret convergence so every experiment trial is set as $T = 100$ iterations. Evaluation shows that acquisition functions are prone to huge loss from devoting all impressions to wrong indices in higher dimensions, where early-stage prediction error is much more likely to occur. The modified Thompson sampling method of Formula 6.2 stands out as optimal policy in predicting ground truth CTRs. Such a sampling method features the flexibility of exploring potentially optimal nodes with low probability even if such nodes are among those with the lowest posterior means, making all-at-once decision mistakes much less likely compared to simpler heuristics of the other policies.

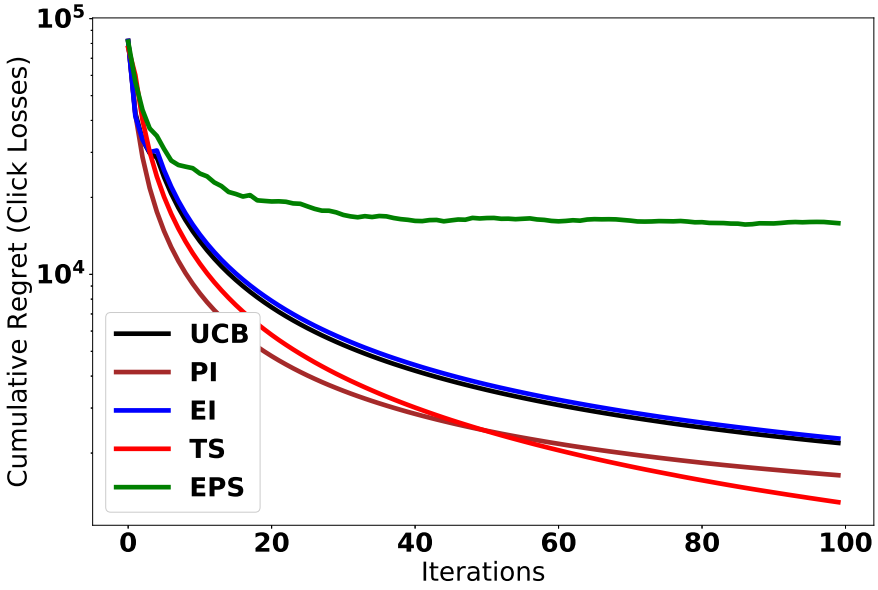



Figure 6.4: 1D Test - Test Bench Environment

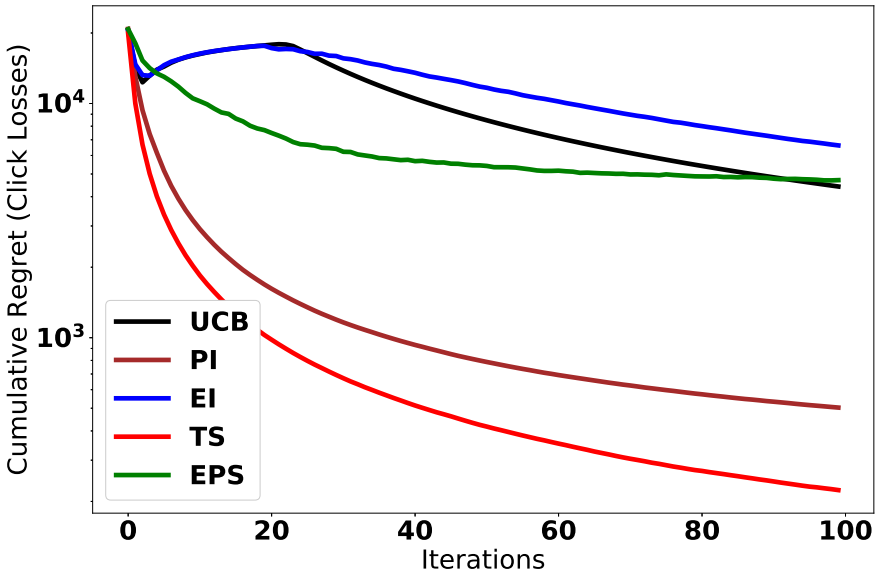



Figure 6.5: 2D Test - Test Bench Environment

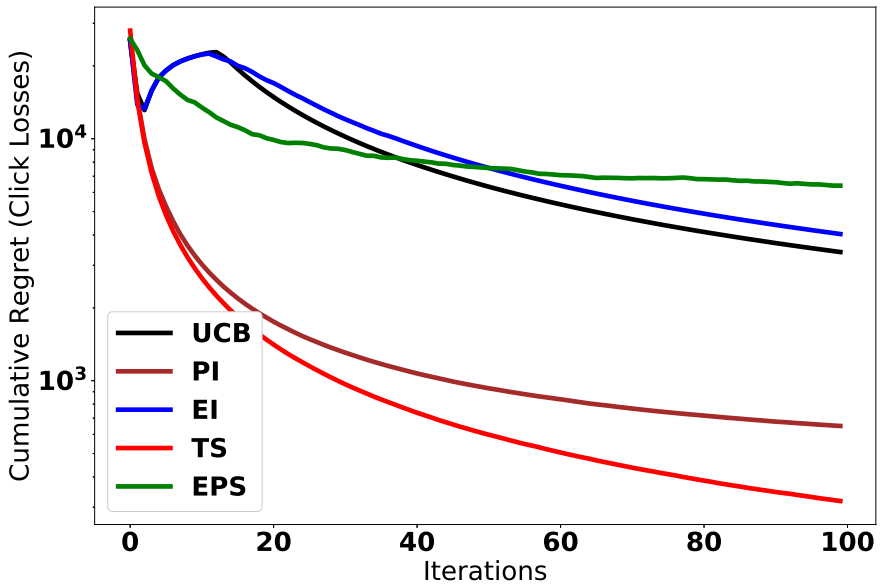



Figure 6.6: 3D Test - Test Bench Environment

## 6.3 Real-life Dataset

The real-life dataset used in this research is a sample of data logs on page views and clicks observed on Websites from different content publishers in the United States during two weeks' time interval. The complete dataset is composed of over 87 million records of page views each of which is in either clicked or unclicked state. All of the logged page views are attributed to 4174 advertisers. For testing the proposed solution in this research, the learning task is designed as a MAB problem of predicting the CTR of content per advertiser. Among the 4174 advertisers the top 100 advertisers with the highest number of page views (impression) are picked up as arms for experiments. These 100 advertisers contribute to more than 37 million page views for the entire dataset.

Figure 6.7 plots the ground truth environment based on the average CTR of the top 1000 advertisers. This environment is treated as a 1D function the predictive model is supposed to learn from. There are two ways this real data environment gets distinguished from testbench environment used in Section 6.1.3. First, the average CTRs in general are much lower than those used in the testbench, which is natural phenomenon as real world ads rarely achieves click likelihood close to 1. Second, distributions of noise vary

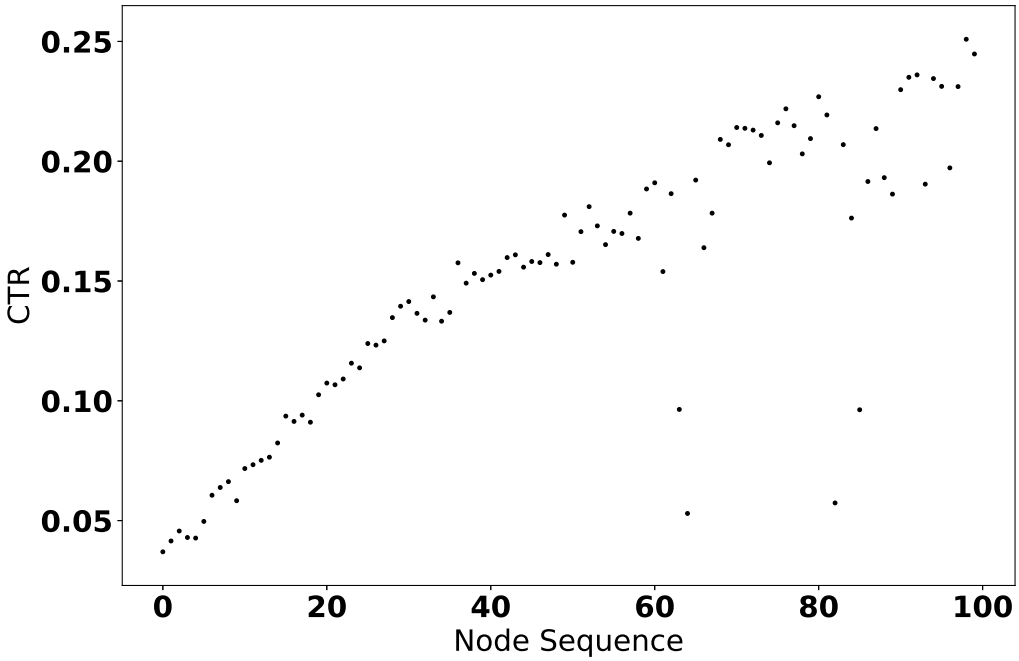


Figure 6.7: Ground Truth Environment for the Dataset

among nodes unlike the i.i.d Gaussian noise manually imposed on synthetic environment in Section 6.1.2. The optimal arm as the figure tells has expected CTR close to 0.25. If the proposed solution, without awareness of the environment, discovers this arm by learning over time the CTR statics purely from the dataset, then the solution succeeds in predicting the optimal advertiser.

The experiment on the real dataset follows virtually the same pattern as in Algorithm 6, except that CTR is collected once every day from the dataset instead of being generated by testbench simulation. Upon the end of a day, the click statistics of all advertisers are collected to compute node CTRs, which are used as sample rewards to fit the predictive model. Again the total budget is set at 100,000 impressions and regret on each day is

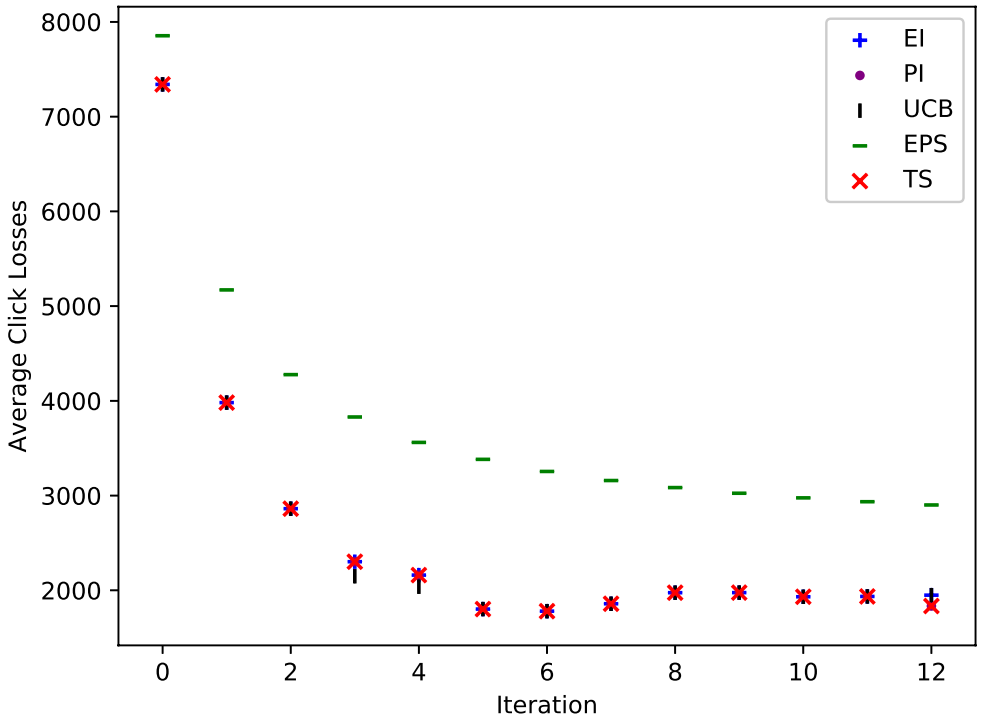



Figure 6.8: Click Loss by Days

computed as click loss compared to choosing the optimal advertiser at CTR of 0.25. Since the time series click logs have a two-week span, the experiment naturally runs 13 iterations so that decisions are always made on the next day after sampling. The same set of decision-making policy applies here. Successful policy is expected to locate the highest CTR and stop exploration after it is discovered. Only in this way does the recommendation traffic get best optimized toward zero regret.

Regret evaluation on the dataset by days is shown in Figure 6.8. Previously evaluated meta-policies present roughly equivalent performance except for naive $\epsilon$-greedy which has a constant exploration factor. Slight fluctuation can be observed on average regret after day 6 and it can be explained by the aforementioned discrepancy of variance among different nodes. As far as the experimented dataset is concerned, data noise is well enough below the tolerant level for the model to finally achieve regret convergence. Still, due

to inconsistency of sample noise among nodes, the modified Thompson sampling technique from Section 6.1.3 takes less advantage of what it assumes to be stationery distributions. Another possible reason of equivalent regret convergence for non-trivial policies is that adjacent nodes of high CTRs are not so well correlated as in the testbench, due to lack of structural learning mentioned at the beginning of Chapter 5. Pre-learning an optimal graph structure is commonly expected for these types of problems GMRF applies to.

Additionally, it is worth noting that in testbench experiments the environment functions of choice are intentionally crafted to have steep global and local optima to make it harder for predictive models to figure out the best index, so that we may rely on the absolute possession of the ground truth $f$ for cumulated regret to be computed bias free. On the contrary, estimating ground truth in real-world application remains an open problem as $f$ cannot be grasped easily. Two common approaches of simulating "golden standard" CTRs include simply averaging [38] CTRs of an ad $i$ as the function value $f(i)$ and excluding rarely clicked items [39] from practical datasets. However, regret calculated from such metrics is prone to underestimation when an action leads to a reward higher than its estimated optimal value in case of extremely noisy environment. For the purpose of this research we focus on verification of model efficacy and consider data wrangling in application dependent scenarios as future work. Therefore ground truth CTRs in the dataset are simplified as average per advertiser.

## 6.4 Further Discussion

### 6.4.1 Sampling Complexity

Through definition by (6.2) this research adopts a novel approach of drawing parameter samples from posterior distributions by following the same semantics of Thompson sampling introduced in Section 5.1.3. However, directly integrating the transcendental function in (6.2) is difficult. The following recipe provides a sequence of more feasible steps of estimating $E[p(y_i >= y_{j \neq i})]$.

$$E[p(y_i >= y_{j\neq i})] = \int_{-\infty}^{\infty} \prod_{j\neq i} \Phi_j(\hat{y}_i)\ d\hat{y}_i \tag{6.4}$$

$$= \sum_{\hat{y}_i} \exp[\sum_{j\neq i} \ln \Phi_j(\hat{y}_i)] \tag{6.5}$$

The inner summation above replaces the product of cumulative density with its exponential equivalent because $\Phi_j(\hat{y}_i)$ are typically small enough for arithmetic underflow to occur when their product is directly calculated. As for range selection of $\hat{y}_i$, a confidence interval of 0.95 is recommended for accurate enough approximation of integral (6.4). Let $\Phi_i(\hat{y}_i|^{0.025}) = 0.025$ and $\Phi_i(\hat{y}_i|^{0.975}) = 0.975$. The numerical estimation of the expected probability of having $y_i$ as an optimal arm out of $n$ arms is expressed as

$$E[p(y_i >= y_{j\neq i})] = \sum_{\hat{y}_i|^{0.005}}^{\hat{y}_i|^{0.975}} \exp[\sum_{j\in[n]\setminus i} \ln \Phi_j(\hat{y}_i)] \tag{6.6}$$

Henceforth the multinomial weight list $\{w_i\}$ is finally computed as

$$w_i = \frac{\sum_{\hat{y}_i|^{0.005}}^{\hat{y}_i|^{0.975}} \exp[\sum_{j\in[n]\setminus i} \ln \Phi_j(\hat{y}_i)]}{\sum_{k=1}^{n} \sum_{\hat{y}_k|^{0.005}}^{\hat{y}_k|^{0.975}} \exp[\sum_{j\in[n]\setminus k} \ln \Phi_j(\hat{y}_k)]} \tag{6.7}$$

Computing $\{w_i\}$ as defined above has cost of $O(kn^2)$ where $k$ is number of integration steps. Experiments in this research takes 0.001 as increment for numerical integration so $k$ is roughly 1000.

### 6.4.2 Numerical Stability

In Experiment 2, computing the integral in (6.2) can potentially lead to numerically unstable issues due to posteriors $y_i$ having a small distribution range $(0, 1)$. This boundary is explained by CTR reward observations $r_i$ (Line 8 in Algorithm 6) because CTR is strictly within $(0, 1)$. Instead of directly sampling observed CTR as $r_i$ we found that Thompson sampling delivers a

much more stable impression weight list if we apply some mapping: $[0, 1] \to \mathbb{R}$ to observations with a logit function so that

$$r_i = \kappa logit(\hat{y}_i) = \ln\big[\frac{\hat{y}_i}{1 - \hat{y}_i}\big] \tag{6.8}$$

The evaluations in Figure 6.4, 6.5 and 6.6 are all conducted using the mapped $r_i$ to update the graphical model.

While both Algorithm 5 and 6 are designed to be fully capable of online learning, real-time sampling may not always be achievable for some performance critical applications such as latency estimation in adaptive routing. Therefore no hard deadline is mandatory in data collection. Specifically, Line 8 in Algorithm 6 can be performed in a sporadic pattern without affecting model inference even if CTR computation is not available during some interval. The test bench used in Experiment 2 only assumes periodic CTR updates because laxity in data stream is irrelevant to the MAB problem itself.

The role of $\kappa$ is a rectifier that increases linearity of the logit function within a subset of its domain enclosing some distance from 0.5. In this research $\kappa$ can be neglected since every experiment mentioned in this thesis takes $\kappa = 1.0$. The setup of (6.8) accommodates cases where reward distributions are close enough to make Thompson sampling difficult to distinguish the optimal and suboptimal arms, due to distribution overlap as explained by (2.2). The low KL-divergence is caused by relatively small derivative of the logit function around 0.5. Choosing a large value of $\kappa$ helps boost the slope of the mapping function near 0.5 and consequently enlarges KL-divergence even when means of two distributions are extremely close to each other.

# Chapter 7

# Concluding Chapter

The overall purpose of designing the proposed solution is to alleviate the cubic complexity $O(m^3)$ nonparametric Gaussian process(GP) suffers from [40] [33] with a training set of size $m$ and such complexity exponentially depends on the dimension of the domain the environment function is defined over. Existing works with similar intention include improving GP performance in high-dimensional spaces with low rank matrix approximation [41] and parallelizable experiment design [42] of bandit optimization under GP setting. Our work differs from existing works by modeling bandit problems with normally distributed rewards from a graphical perspective instead of extending techniques in GP regression. Similar to GP, the proposed model predicts unseen distributions based on seen samples that are collected with certain noise. But our model notably differs from GP by adhering to discrete space instead of continuous function domain. This is because fully interpolating a continuous function generally overkills the bandit problem in case of a finite number of arms. Consequently, we restrict the computational cost of inferring posterior distributions to $O(n^3)$ with a graph of $n$ nodes independent from the size of training samples. The proposed model is thus free from penalty of increasing complexity as the observation set grows. Comprehensive performance comparison between the graphical model and GP model is listed in Table 7.1.

As previously stated no rigid parametric assumption is made when our predictive model learns the "black box" function. However we do suggest that at least some degree of smoothness be granted so that the precision matrix can be best potentiated. As with most stochastic bandit problems

| | GP | GMRF |
|---|---|---|
| Sampling Space | continuous | discrete |
| Model Size Complexity | $O(m)$ | $O(1)$ |
| Time Complexity (Inference) | $O(m^3)$ | $O(n^3)$ |
| Scalability | poor to training data | fine to training data<br>poor to feature space size |

Table 7.1: Pros and Cons between GP and GMRF

this paper assumes stationery environment only subject to certain noise and considers that the ground truth is never mutated by actions that have been taken, which otherwise constitutes a full interactive reinforcement learning problem. Since the proposed model is merely an abstraction that is applicable to any finite discrete space of random variables, it does not theoretically precludes contextual bandit setting. Still, in this work we do not expect any generalization to contextual bandits due to exploding action space, typically defined by the cross product of the function index space and context space. Efficiently retrieving optimal rewards in contextual bandit problems has been persistently studied. The major difficulty lies within the complexity of interpreting contextual information. Commonly used approaches include directly clustering context vectors [43], using hierarchical context feature representation [44] for efficient exploration and combining solutions to the MAB problem with similarity based ranking algorithm to select subsets [45] within item-context space.

## 7.1 The Bandit Problem and Markov Decision Process

A Markov decision process (MDP) is a stochastic discrete decision-making problem over a non-empty finite set of states. A MDP problem [46] is formulated as a triplet of $(\mathbb{S}, \mathbb{A}, \mathbb{P})$, standing for a state set, action set and transition probability kernel. Transition probability $\mathcal{P} \in \mathbb{P}$ assigns a state-action pair $(s, a), s \in \mathbb{S}, a \in \mathbb{A}$ a probability distribution table in which $\mathcal{P}(s, a, s')$ is the probability of transitioning state $s$ into $s'$ after action $a$ is taken. For every transition $(s, a, s')$ there is corresponding *immediate reward* $R_{s,a,s'}$. Then a

reward function $r : \mathbb{S} \times \mathbb{A} \to \mathbb{R}$ can be defined as

$$r(s, a) = \sum_{s'} \mathcal{P}(s'|a, s) R_{s,a,s'}$$

where $\sum_{s'} \mathcal{P}(s'|a, s) = 1$. The MAB problem is essentially a particular case of MDP. In stochastic bandit setting, only one state is available $\mathbb{S} = \{\mathcal{S}_0\}$ and $\mathcal{S}_0$ is constantly considered as the terminal state. Notion of state transition $\mathcal{P}$ is then naturally stripped off turning the reward function into a much simplified form $r(a)$, because any every action determinately leads to the terminal state.

The probability distributions behind the simplified reward function $r(a)$ are the exact list of arm rewards $[X_1, X_2, ...X_k]$ where $a \in [k]$. This implies that solutions to the MAB problem can be useful to local optimization in MDP given that $r(s, a)$ has stationery distribution.

## 7.2 The Bandit Problem and Reinforcement Learning

A reinforcement learning (RL) problem models unknown environment in which MDP takes place. The agency, who is responsible for making decisions trying to collect a maximum amount of rewards is expected to learn environment property without prior knowledge purely through experience, in a similar pattern to the MAB problem. It usually takes two steps for a MAB problem to get generalized to a full-feature reinforcement learning problem. First, consider introducing a state space $\mathcal{S}$ the MAB problem and extending its reward $r(a)$ into $r(s, a)$. In this way, one-step rewards not only depend on choice of arms $a$ but also get affected by the current state. In other words, the MAB problem is not turned into a *contextual multi-armed bandit problem* which states that given some context $s \in \mathcal{S}$ and action space $\mathcal{A}$, the agency is expected to learn and predict which action maximizes the reward under given context. Table 7.2 lists some high-level difference between the MAB the CMAB problem in terms of how reward is measured. Since context is the only extra parameter introduced to CMAB reward function, learning CMAB policy can break down to state specific decision making, i.e., an optimal arm is to be found separately for every possible state $s \in \mathcal{S}$.

| | states | policy | reward | action-state impact |
|---|---|---|---|---|
| MAB | (stateless) | $\pi$,$\pi \in \mathbb{A}$ | $r(a)$ | ✗ |
| CMAB | $\mathbb{S}$ | $\pi(s)$,$s \in \mathbb{S}$ | $r(s,a)$ | ✗ |
| RL | $\mathbb{S}$ | $\pi(a\|s)$,$s \in \mathbb{S}$, $a \in \mathbb{A}$ | $\int_{s'} r(s,a)ds'$ | ✓ |

Table 7.2: Bandit and Reinforcement Learning Problems

Both MAB and CMAB problems come with considerably strong assumption that no environment feedback is received after an action is taken, compared to a RL problem in which an action may alter the state of the agency. Such impact by action requires redefinition of immediate reward because reward is no longer bound to specific state-action pair when state transition is considered. The expected immediate reward listed in Table 7.2 is one commonly used approach of estimating immediate reward, known as *Monte Carlo* method because it averages immediate rewards over random transitions. The learning objective also differs in RL problems. In bandit problems, policy produces an index that is either a scalar or a parameterized integer indicating the optimal arm. Learning optimal policy in a RL problem is much more complicated because a deterministic policy function is hardly sufficient for maximizing the long term reward. Instead an agent in RL environment commonly learns a distribution of actions for individual states, as is reflected in Table 7.2.

Most policy learning algorithms in RL problems fall into two categories: model-based methods that attempt to estimate transition details of MDP from environment and model-free methods that ignore MDP and directly approximate an long term reward function (return function). Regardless of which category is applied, a RL agent relies on estimation of immediate rewards to achieve global policy that is long term optimal. Hereby solving bandit problems is helpful to improving local correctness of immediate rewards, which are crucial to solving many RL problems.

## 7.3 Summary

This dissertation reviews nature of the multi-armed bandit problem, summarizes a few typical existing solutions and proposes a novel abstract model

for solving this problem in a cost-aware approach. The proposed model is based upon Gaussian Markov random field for learning from sparse data samples and predicting reward distributions as function outputs at discrete indices. Predictions optimize the multi-armed bandit problem at best achievable cost under a variant of Thompson sampling as decision making policy. Experiments illustrate that the designated algorithm helps reduce online cost and achieves satisfactory loss convergence under budget constraint in both synthetic Gaussian environment and real-life scenarios of binomial click patterns. Therefore our solution is applicable to profit improvement for recommendation engines as well as other online marketing scenarios that demand seeking for optimum from unknown environment. Future works include more in-depth study on more efficient computation of graphical inference and scalability issues towards more practical use cases including contextual bandit problems.

# Reference


[1] Christian, B., Griffiths, T.: Algorithms to Live By: The Computer Science of Human Decisions. Henry Holt and Co., Inc., New York, NY, USA (2016)

[2] Wasserman, L.: All of Statistics: A Concise Course in Statistical Inference. Springer Publishing Company, Incorporated (2010)

[3] Kaufmann, E., Cappe, O., Garivier, A.: On Bayesian Upper Confidence Bounds for Bandit Problems. Proceedings of Machine Learning Research, La Palma, Canary Islands, PMLR (21–23 Apr 2012) 592–600

[4] Lai, T., Robbins, H.: Asymptotically efficient adaptive allocation rules. Adv. Appl. Math. **6**(1) (March 1985) 4–22

[5] Foucart, S., Rauhut, H.: A Mathematical Introduction to Compressive Sensing. Birkhäuser Basel (2013)

[6] Sutton, R.S., Barto, A.G.: Introduction to Reinforcement Learning. 1st edn. MIT Press, Cambridge, MA, USA (1998)

[7] Kuleshov, V., Precup, D.: Algorithms for Multi-armed Bandit Problems. CoRR **abs/1402.6028** (2014)

[8] White, J.: Bandit Algorithms for Website Optimization: Developing, Deploying, and Debugging. O'Reilly Media (2012)

[9] Auer, P., Cesa-Bianchi, N., Fischer, P.: Finite-time analysis of the multiarmed bandit problem. Machine Learning **47**(2) (May 2002) 235–256

[10] Wang, Y., yves Audibert, J., Munos, R.: Algorithms for infinitely many-armed bandits. In Koller, D., Schuurmans, D., Bengio, Y., Bottou, L., eds.: Advances in Neural Information Processing Systems 21. Curran Associates, Inc. (2009) 1729–1736

[11] Russo, D.J., Van Roy, B., Kazerouni, A., Osband, I., Wen, Z., et al.: A Tutorial on Thompson Sampling. Foundations and Trends in Machine Learning **11**(1) (2018) 1–96

[12] Srinivas, N., Krause, A., Kakade, S., Seeger, M.: Gaussian process optimization in the bandit setting: No regret and experimental design. In: Proceedings of the 27th International Conference on International Conference on Machine Learning. ICML'10, USA, Omnipress (2010) 1015–1022

[13] Rasmussen, C.E., Williams, C.K.I.: Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning). The MIT Press (2005)

[14] Cover, T.M., Thomas, J.A.: Elements of Information Theory. Wiley-Interscience, New York, NY, USA (1991)

[15] Krause, A., Guestrin, C.: Near-optimal nonmyopic value of information in graphical models. In: Proceedings of the Twenty-First Conference on Uncertainty in Artificial Intelligence. UAI'05, Arlington, Virginia, United States, AUAI Press (2005) 324–331

[16] Nemhauser, G.L., Wolsey, L.A., Fisher, M.L.: An analysis of approximations for maximizing submodular set functions. Mathematical Programming **14**(1) (1978) 265–294

[17] Koller, D., Friedman, N.: Probabilistic Graphical Models: Principles and Techniques - Adaptive Computation and Machine Learning. The MIT Press (2009)

[18] Lafferty, J.D., McCallum, A., Pereira, F.C.N.: Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In: Proceedings of the Eighteenth International Conference on Machine Learning. ICML '01, San Francisco, CA, USA, Morgan Kaufmann Publishers Inc. (2001) 282–289

[19] Dechter, R., Rish, I.: Mini-buckets: A general scheme for bounded inference. Journal of the ACM (JACM) **50**(2) (2003) 107–153

[20] Dechter, R.: Bucket elimination: A unifying framework for reasoning. Artificial Intelligence **113**(1-2) (1999) 41–85

[21] Dechter, R.: Reasoning with probabilistic and deterministic graphical models: Exact algorithms. Synthesis Lectures on Artificial Intelligence and Machine Learning **7**(3) (2013) 1–191

[22] Zeng, C., Wang, Q., Mokhtari, S., Li, T.: Online context-aware recommendation with time varying multi-armed bandit. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM (2016) 2025–2034

[23] Nguyen, T.V., Karatzoglou, A., Baltrunas, L.: Gaussian process factorization machines for context-aware recommendations. In: Proceedings of the 37th international ACM SIGIR conference on Research & development in information retrieval, ACM (2014) 63–72

[24] Sui, Y., Gotovos, A., Burdick, J.W., Krause, A.: Safe exploration for optimization with gaussian processes. In: Proceedings of the 32Nd International Conference on International Conference on Machine Learning - Volume 37. ICML'15, JMLR.org (2015) 997–1005

[25] Schreiter, J., Nguyen-Tuong, D., Eberts, M., Bischoff, B., Markert, H., Toussaint, M.: Safe exploration for active learning with gaussian processes. In: Proceedings of the 2015th European Conference on Machine Learning and Knowledge Discovery in Databases - Volume Part III. ECMLPKDD'15, Switzerland, Springer (2015) 133–149

[26] Bubeck, S., Munos, R., Stoltz, G.: Pure exploration in multi-armed bandits problems. In: International conference on Algorithmic learning theory, Springer (2009) 23–37

[27] Rue, H., Held, L.: Gaussian Markov Random Fields: Theory And Applications (Monographs on Statistics and Applied Probability). Chapman & Hall/CRC (2005)

[28] Wu, Y., György, A., Szepesvári, C.: Online learning with gaussian payoffs and side observations. In: Advances in Neural Information Processing Systems. (2015) 1360–1368

[29] Chapelle, O., Li, L.: An empirical evaluation of thompson sampling. In: Proceedings of the 24th International Conference on Neural Information Processing Systems. NIPS'11, USA, Curran Associates Inc. (2011) 2249–2257

[30] Agrawal, S., Goyal, N.: Thompson sampling for contextual bandits with linear payoffs. In: International Conference on Machine Learning. (2013) 127–135

[31] Honda, J., Takemura, A.: An asymptotically optimal policy for finite support models in the multiarmed bandit problem. Machine Learning **85**(3) (2011) 361–391

[32] Maes, F., Wehenkel, L., Ernst, D.: Learning to play k-armed bandit problems. In: Proceedings of the 4th International Conference on Agents and Artificial Intelligence (ICAART 2012). (2012)

[33] Wang, Z., Zhou, B., Jegelka, S.: Optimization as estimation with gaussian processes in bandit settings. In: Artificial Intelligence and Statistics. (2016) 1022–1031

[34] Kushner, H.J.: A new method of locating the maximum point of an arbitrary multipeak curve in the presence of noise. Journal of Basic Engineering **86**(1) (1964) 97–106

[35] Močkus, J.: On bayesian methods for seeking the extremum. In: Optimization Techniques IFIP Technical Conference, Springer (1975) 400–404

[36] Jones, D.R., Schonlau, M., Welch, W.J.: Efficient global optimization of expensive black-box functions. Journal of Global optimization **13**(4) (1998) 455–492

[37] Srinivas, N., Krause, A., Kakade, S.M., Seeger, M.W.: Information-theoretic regret bounds for gaussian process optimization in the bandit setting. IEEE Transactions on Information Theory **58**(5) (2012) 3250–3265

[38] Mary, J., Preux, P., Nicol, O.: Improving offline evaluation of contextual bandit algorithms via bootstrapping techniques. In: International Conference on Machine Learning. (2014) 172–180

[39] Wang, X., Li, W., Cui, Y., Zhang, R., Mao, J.: Click-through rate estimation for rare events in online advertising. In: Online Multimedia Advertising: Techniques and Technologies. IGI Global (2011) 1–12

[40] Rasmussen, C., Williams, C.: Gaussian Processes for Machine Learning. Adaptative computation and machine learning series. University Press Group Limited (2006)

[41] Djolonga, J., Krause, A., Cevher, V.: High-dimensional gaussian process bandits. In: Advances in Neural Information Processing Systems. (2013) 1025–1033

[42] Desautels, T., Krause, A., Burdick, J.W.: Parallelizing exploration-exploitation tradeoffs in gaussian process bandit optimization. The Journal of Machine Learning Research **15**(1) (2014) 3873–3923

[43] Li, L., Chu, W., Langford, J., Schapire, R.E.: A contextual-bandit approach to personalized news article recommendation. In: Proceedings of the 19th international conference on World wide web, ACM (2010) 661–670

[44] Yue, Y., Hong, S.A., Guestrin, C.: Hierarchical exploration for accelerating contextual bandits. In: Proceedings of the 29th International Coference on International Conference on Machine Learning. ICML'12, USA, Omnipress (2012) 979–986

[45] Vanchinathan, H.P., Nikolic, I., De Bona, F., Krause, A.: Explore-exploit in top-n recommender systems via gaussian processes. In: Proceedings of the 8th ACM Conference on Recommender systems, ACM (2014) 225–232

[46] Szepesvári, C.: Algorithms for Reinforcement Learning. Synthesis lectures on artificial intelligence and machine learning. Morgan & Claypool (2010)

# 著者研究業績目録

## 査読付き論文雑誌

1. Zhao Chen, Bin Yang, and Yu Hirate. A reward optimization model for decision-making under budget constraint. 情報処理学会論文誌データベース（TOD）, Vol. 12, No. 1, January 2019.

## 査読付き国際会議論文

1. Takakazu Imada, Yusuke Inoue, Lei Chen, Syunya Doi, Tian Nie, Chen Zhao, Takehito Utsuro, and Yasuhide Kawada. Analyzing time series changes of correlation between market share and concerns on companies measured through search engine suggests. In *Proceedings of the 10th International Conference on Language Resources and Evaluation*, pp. 1917–1923, May 2016.

2. Tian Nie, Yi Ding, Chen Zhao, Youchao Lin, Takehito Utsuro, and Yasuhide Kawada. Clustering search engine suggests by integrating a topic model and word embeddings. In *Proceedings of the 18th IEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing*, pp. 581–586, June 2017.

3. Jiaqi Li, Chen Zhao, Youchao Lin, Mizuho Baba, Tian Nie, Takehito Utsuro, Yasuhide Kawada, and Noriko Kando. A method of collecting know-how knowledge based on question-answer examples and search engine suggests. In *Proceedings of the 11th International Conference on Ubiquitous Information Management and Communication*, January 2017.

4. Chen Zhao, Jiaqi Li, Tian Nie, Yi Ding, Linghan Xu, Takehito Utsuro, Yasuhide Kawada, and Noriko Kando. Identifying major contents among Web pages with search engine suggests by modeling topics. In *Proceedings of the 11th International Conference on Ubiquitous Information Management and Communication*, January 2017.

5. Chen Zhao, Kohei Watanabe, Bin Yang, and Yu Hirate. Fast converging multi-armed bandit optimization using probabilistic graphical model. In D. Phung,

V. Tseng, G. Webb, B. Ho, M. Ganji, and L. Rashidi, editors, *Advances in Knowledge Discovery and Data Mining — 22nd Pacific-Asia Conference, PAKDD 2018, Melbourne, VIC, Australia, June 3-6, 2018, Proceedings, Part II*, Vol. 10938 of *Lecture Notes in Computer Science*, pp. 115–127. Springer, June 2018.

# 報告

1. 聶添, 徐凌寒, 趙辰, 宇津呂武仁, 河田容英. サジェストおよびトピックモデルを用いた多様な話題のウェブページの選択的収集. 第 30 回人工知能学会全国大会論文集, June 2016.
2. 宇津呂武仁, 徐凌寒, 聶添, 趙辰, 李佳奇, 河田容英. 企業名に関する関心動向のトピックモデリングを用いた日中市場シェアの分析. 第 30 回人工知能学会全国大会論文集, June 2016.
3. 趙辰, 丁易, 聶添, 李佳奇, 宇津呂武仁, 河田容英, 神門典子. サジェストおよびトピックモデルを用いた話題集約における主要話題ページ同定. 第 9 回データ工学と情報マネジメントに関するフォーラム—DEIM フォーラム— 論文集, March 2017.
4. 李佳奇, 趙辰, 林友超, 馬場瑞穂, 宇津呂武仁, 河田容英, 神門典子. トピックモデルにおける話題分布特性を手がかりとするノウハウサイトの収集. 第 9 回データ工学と情報マネジメントに関するフォーラム—DEIM フォーラム— 論文集, March 2017.
5. 聶添, 丁易, 趙辰, 李佳奇, 宇津呂武仁, 河田容英. トピックモデルおよび分散表現の併用による検索エンジン・サジェストの集約. 第 31 回人工知能学会全国大会論文集, May 2017.
6. Jiaqi Li, Shuto Kawabata, Chen Zhao, Yi Ding, Youchao Lin, Takehito Utsuro, and Yasuhide Kawada. Collecting know-how sites based on search engine suggests and a topic model. In *the 17th China-Japan Natural Language Processing Joint Research Promotion Conference*, September 2017.
7. 李佳奇, 趙辰, 林友超, 丁易, 川畑修人, 宇津呂武仁, 河田容英. トピックモデルおよび分類器学習を用いたノウハウサイトの同定. 第 10 回データ工学と情報マネジメントに関するフォーラム—DEIM フォーラム— 論文集, March 2018.
8. 丁易, 趙辰, 川畑修人, 宇津呂武仁, 河田容英. トピックモデル・分散表現の併用によるウェブ検索結果話題集約におけるサブトピック化. 第 10 回データ工学と情報マネジメントに関するフォーラム—DEIM フォーラム— 論文集, March 2018.