Electronic Preprint for Journal of Information Processing Vol.27

Regular Paper

A Reward Optimization Model for Decision-making under Budget Constraint

Chen Zhao[1,a)] Bin Yang[2] Yu Hirate[2]

Received: June 2, 2018, Accepted: October 5, 2018

Abstract: This paper designs 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. While properly trained neural networks are 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. An online algorithm is also presented to show how model inference is used to locate global optima of an unknown function, as the primary objective of making optimal decisions. Comparative experiments are conducted among a set of sampling policies to demonstrate how click-through rates can be improved by optimized recommendation strategy with the proposed model. Empirical evaluation suggests that an adapted version of Thompson sampling is the best suitable policy for the proposed algorithm.

Keywords: reinforcement learning, probabilistic graphical model, multi-armed bandit, recommendation system

1. Introduction

This paper 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 [10], [22]. 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 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 paper models this initialization problem as reward optimization with exploration-exploitation trade-off under the paradigm of the multi-armed bandit (MAB) problems [2], [13], [16] 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.

Primary contribution of this paper is a two-fold solution. The first part is a 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) [12], 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

[1] Graduate School of System and Information Engineering, University of Tsukuba, Tsukuba, Ibaraki 305–0005, Japan
[2] Rakuten Institute of Technology, Setagaya, Tokyo 158–0094, Japan
[a)] s1630190@u.tsukuba.ac.jp

This paper is an extended version of our previous work [23] presented in the 22nd Pacific-Asia Conference on Advances in Knowledge Discovery and Data Mining, PAKDD 2018.

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 [20] 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 part of our solution is sampling-based decision making policies. As graphical model inference provides the posterior of item rewards, determining the best candidate item is broken down into a multi-armed bandit problem. The proposed decision making policy is a variation of Thompson sampling [1], [3] 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 the paper is structured as follows. First the graphical model is defined along with the inference process with proof on its GMRF property and correctness of probability computation. Second learning process is introduced as on-line algorithm in which decision making and model adjustments take place in parallel. The algorithm is experimented on both synthetic environment and real CTR test benches. Discussion of experiment results comes right before conclusion.

2. Model Declaration

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 ϵ 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 $\mathbb{S}$.

$$\mathbb{S} = \{(i, r_i) \mid r_i = f(i) + \epsilon(i), i \in D^d\} \tag{1}$$

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.

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.

2.2.1 Markov property

Graph construction starts with Markov property among nodes y_i as presented in the example Markov random field by **Fig. 1**,

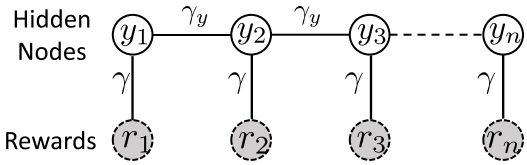

Fig. 1 Markov property.

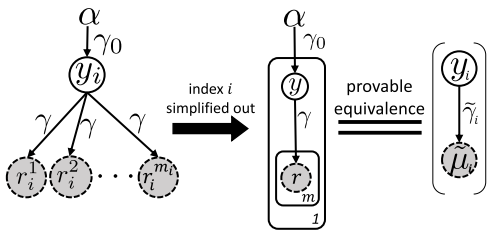

Fig. 2 Bayesian property.

where every y_i stands for a *hidden node* or *target node* that infers probability distribution of r_i defined in Eq. (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 γ_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.

$$p(\boldsymbol{y}|\gamma_y) = \prod_{i=1}^{n-1} \exp\left(-\frac{\gamma_y}{2}(y_{i+1} - y_i)^2\right) \tag{2}$$

$$= \prod_{i,j \in E} \exp\left(-\frac{\gamma_y}{2}(y_i - y_j)^2\right) \tag{3}$$

In general cases, graph nodes y_i are not necessarily linearly connected as in the particular example of Fig. 1. A more comprehensive expression of Markov joint density is expressed as Eq. (3) in clique factors from a pairwise Markov network with E being the edge set.

2.2.2 Bayesian property

Setup of the Markov network in Fig. 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 2** denotes the Bayesian network that expresses such property. Under Bayesian assumption, every target node y_i is conditioned on some prior α with bonding strength γ_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

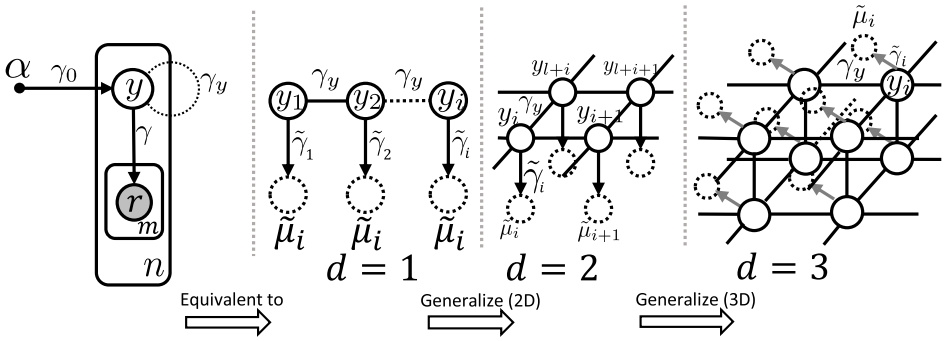

Fig. 3 Final Graphical Model - Gaussian Markov Random Field.

conditionally independent given their hidden node y_i. Similar to the Markov network, bonding coefficient γ 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.

$$p(\boldsymbol{r}^{(i)}, y_i|\gamma, \gamma_0, \alpha) = p(y_i|\gamma_0, \alpha) \prod_{j=1}^{m_i} p(r_j^{(i)}|\gamma, y_i) \tag{4}$$

In Eq. (4), $\boldsymbol{r}^{(i)}$ is the reward list of y_i and $r_j^{(i)}$ is the jth sample. Reward list sizes m_i are expected to vary among target nodes. Computing the product of likelihood over observations using Eq. (4) 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 μ_i of $\boldsymbol{r}^{(i)}$ so as to eliminate child indices of y_i.

$$p(\boldsymbol{r}^{(i)}, y_i|\gamma, \gamma_0, \alpha) = p(y_i|\gamma_0, \alpha)[p(\mu_i|\gamma, y_i)]^{m_i} \tag{5}$$

$$= \exp\left[-\frac{1}{2}\gamma_0(y_i - \alpha)^2\right] \exp\left[-\frac{m_i}{2}\gamma(y_i - \mu_i)^2\right] \tag{6}$$

$$\text{where } \mu_i = \left(\sum_{j=1}^{m_i} r_j^{(i)}\right) \Big/ m_i \tag{7}$$

Similar to Eq. (3), joint density between connected nodes are modeled with Gaussian functions subject to bonding strength γ_0, γ in Eq. (6). We further work on Eq. (6) by expanding all the terms in the exponential part.

$$\begin{aligned}
&p(\boldsymbol{r}^{(i)}, y_i|\gamma, \gamma_0, \alpha) \qquad (8)\\
&= \exp\left\{-\frac{1}{2}[m_i\gamma(y_i - \mu_i)^2 + \gamma_0(y_i - \alpha)^2]\right\}\\
&= \exp\left\{-\frac{1}{2}\left[(m_i\gamma + \gamma_0)y_i^2 - (2\gamma m_i\mu_i + 2\gamma_0\alpha)y_i + \gamma_0\alpha^2 + m_i\gamma\mu_i^2\right]\right\}\\
&= \exp\left\{-\frac{1}{2}(m_i\gamma + \gamma_0)\left[y_i^2 - \frac{2\gamma m_i\mu_i + 2\gamma_0\alpha}{m_i\gamma + \gamma_0}y_i + \frac{\gamma_0\alpha^2 + m_i\gamma\mu_i^2}{\gamma_0 + m_i\gamma}\right]\right\}\\
&= \exp\left\{-\frac{1}{2}(m_i\gamma + \gamma_0)\left[y_i^2 - \frac{2\gamma m_i\mu_i + 2\gamma_0\alpha}{m_i\gamma + \gamma_0}y_i + C\right]\right\}
\end{aligned}$$

Expression (8) 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.

$$p(\boldsymbol{r}^{(i)}, y_i|\gamma, \gamma_0, \alpha) = \exp\left\{-\frac{1}{2}\tilde{\gamma}_i(y_i - \tilde{\mu}_i)^2 + C'\right\} \tag{9}$$

$$= \exp\{C'\} \exp\left\{-\frac{1}{2}\tilde{\gamma}_i(y_i - \tilde{\mu}_i)^2\right\} \tag{10}$$

$$\text{where } \tilde{\gamma}_i = m_i\gamma + \gamma_0 \text{ and } \tilde{\mu}_i = \frac{\gamma m_i\mu_i + \gamma_0\alpha}{m_i\gamma + \gamma_0} \tag{11}$$

Expression (11) 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 values are μ_i rescaled by γ, γ_0 and α. Also notice that $\tilde{\gamma}_i$ differs from constant γ since it depends on reward list sizes as given in Eq. (11). The finally simplified Bayesian network is also displayed in Fig. 2.

2.2.3 Final representation

The final graphical model as **Fig. 3** is a consolidated graph with every hidden node component in Fig. 1 replaced by the simplified Bayesian representation in Fig. 2. The plate notation shows an example of n hidden nodes that are bonded with γ_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 Eq. (9) on target nodes and Markov joint density counting every edge using Eq. (3).

$$p(\boldsymbol{y}, \boldsymbol{r}|\alpha, \gamma_y, \gamma_0, \gamma) = p(\boldsymbol{y}|\gamma_y) \prod_i \left[p(\boldsymbol{r}^{(i)}, y_i|\gamma, \gamma_0, \alpha)\right] \tag{12}$$

To make Eq. (12) explicit, we now plug in both probability factors from Eq. (10) and Eq. (3) ignoring the constant factor.

$$\begin{aligned}
&p(\boldsymbol{y}, \boldsymbol{r}|\alpha, \gamma_y, \gamma_0, \gamma) \propto\\
&\left\{\prod_i \exp\left[-\frac{1}{2}\tilde{\gamma}_i(y_i - \tilde{\mu}_i)^2\right]\right\} \left\{\prod_{i,j\in E} \exp\left[-\frac{\gamma_y}{2}(y_i - y_j)^2\right]\right\} \qquad (13)
\end{aligned}$$

In Eq. (13) 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 Eq. (1). A typical GMRF node y_i has an increasing count of neighbors as dimension d goes up as is shown in Fig. 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 Eq. (13), 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.

$$\ln p(\boldsymbol{y}, \boldsymbol{r}|\alpha, \gamma_y, \gamma_0, \gamma) \propto \sum_i \left\{-\frac{1}{2}\tilde{\gamma}_i(y_i - \tilde{\mu}_i)^2\right\} + \sum_{i,j\in E}\left\{-\frac{\gamma_y}{2}(y_i - y_j)^2\right\} \quad (14)$$

3. Graphical Inference

This section discusses, on top of the proposed graphical model, how inference is conducted. Now that Eq. (14) 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 α, γ_y, γ_0, γ 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.

$$E(\boldsymbol{y}) = \sum_i \left\{\tilde{\gamma}_i(y_i - \tilde{\mu}_i)^2\right\} + \sum_{i,j\in E}\left\{\gamma_y(y_i - y_j)^2\right\} \quad (15)$$

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 γ_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 xth 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 xth diagonal element.

$$\boldsymbol{A} = \boldsymbol{B} - \boldsymbol{K} + \gamma_y diag(\boldsymbol{k}) = \begin{pmatrix} \gamma_y k_1 + \tilde{\gamma}_1 & -\gamma_y & \cdots & -\gamma_y & \cdots & & \\ -\gamma_y & \gamma_y k_2 + \tilde{\gamma}_2 & \cdots & \cdots & -\gamma_y & & \\ \vdots & \vdots & \ddots & & & & \\ -\gamma_y & \vdots & & \ddots & & & \\ & -\gamma_y & & & \ddots & & \\ & & \cdots & & & \gamma_y k_n + \tilde{\gamma}_n \end{pmatrix} \quad (16)$$

Having matrix A and B explicitly defined, we now exploit a provable fact (proof given in Appendix A.1) that $E(\boldsymbol{y})$ is equivalent to product of the following nested matrices.

$$E(\boldsymbol{y}) = \begin{pmatrix}\boldsymbol{y}^\top & \tilde{\boldsymbol{\mu}}^\top\end{pmatrix}\begin{pmatrix}\boldsymbol{A} & -\boldsymbol{B} \\ -\boldsymbol{B} & \boldsymbol{B}\end{pmatrix}\begin{pmatrix}\boldsymbol{y} \\ \tilde{\boldsymbol{\mu}}\end{pmatrix} \quad (17)$$

$$= \boldsymbol{y}^\top\boldsymbol{A}\boldsymbol{y} - \tilde{\boldsymbol{\mu}}^\top\boldsymbol{B}\boldsymbol{y} - \boldsymbol{y}^\top\boldsymbol{B}\tilde{\boldsymbol{\mu}} + \tilde{\boldsymbol{\mu}}^\top\boldsymbol{B}\tilde{\boldsymbol{\mu}} \quad (18)$$

Recall that $\boldsymbol{y}$ is the target node list and $\tilde{\boldsymbol{\mu}}$ is the interpolated sample means of target nodes. $\begin{pmatrix}\boldsymbol{y}^\top & \tilde{\boldsymbol{\mu}}^\top\end{pmatrix}$ 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 (17) produces a product of three matrices of sizes $1 \times 2n$, $2n \times 2n$ and $2n \times 1$.

The optimal $\hat{\boldsymbol{y}}$ is $\boldsymbol{y}$ that minimizes $E(\boldsymbol{y})$ and equivalently maximizes the model probability. There exists a provable closed-form solution (proof given in Appendix A.2) of $\hat{\boldsymbol{y}}$ defined with matrix notation below.

$$\hat{\boldsymbol{y}} = \underset{\boldsymbol{y}}{\operatorname{argmax}}\left(\log p(\boldsymbol{r}, \boldsymbol{y}|\gamma_y, \gamma, \gamma_0, \alpha)\right) \quad (19)$$

$$= \boldsymbol{A}^{-1}\boldsymbol{B}\tilde{\boldsymbol{\mu}} \text{ and } \boldsymbol{\sigma}^2_{\hat{\boldsymbol{y}}} = diag(\boldsymbol{A}^{-1}) \quad (20)$$

$\boldsymbol{\sigma}^2_{\hat{\boldsymbol{y}}}$ 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

$$p(\boldsymbol{r}, \boldsymbol{y}|\gamma, \gamma_y, \gamma_0, \alpha) \propto \exp\left(-\frac{1}{2}\tilde{\boldsymbol{\mu}}^\top\Lambda\tilde{\boldsymbol{\mu}}\right) \quad (21)$$

Formula (21) proves that the final graph in Fig. 3 conforms to GMRF property with multivariate Gaussian distribution such that $p(\boldsymbol{y}) \sim \mathcal{N}\left(\tilde{\boldsymbol{\mu}}, \Lambda^{-1}\right)$. Λ serves as the precision matrix.

Lastly, computing $\boldsymbol{A}$ defined in Eq. (16) 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.

$$\boldsymbol{A'} = \boldsymbol{B} - \boldsymbol{K} + 2d\gamma_y\boldsymbol{I}_n = \begin{pmatrix} 2d\gamma_y + \tilde{\gamma}_1 & -\gamma_y & \cdots & -\gamma_y & \cdots & & \\ -\gamma_y & 2d\gamma_y + \tilde{\gamma}_2 & \cdots & \cdots & -\gamma_y & & \\ \vdots & \vdots & \ddots & & & & \\ -\gamma_y & \vdots & & \ddots & & & \\ & -\gamma_y & & & \ddots & & \\ & & \cdots & & & 2d\gamma_y + \tilde{\gamma}_n \end{pmatrix} \quad (22)$$

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 experiments in this paper.

$$\hat{\boldsymbol{y}} = \underset{\boldsymbol{y}}{\operatorname{argmax}}\left(\log p(\boldsymbol{r}, \boldsymbol{y}|\gamma_y, \gamma, \gamma_0, \alpha)\right) = \boldsymbol{A'}^{-1}\boldsymbol{B}\tilde{\boldsymbol{\mu}} \quad (23)$$

$$\boldsymbol{\sigma}^2_{\hat{\boldsymbol{y}}} = diag(\boldsymbol{A'}^{-1}) \quad (24)$$

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

4. Brief Review on the Multi-armed Bandit

The multi-armed bandit (MAB) is a widely studied [6], [8], [19] decision making problem associated with environment reward. 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 μ_i as mean of Y_i, a policy decides the index $\pi(t)$ of the chosen arm at the tth step and $r_i^{(t)}$ is the sample reward observed on the ith 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 μ_i in MAB are never accessible so μ^* can only be approximated through inference. This paper models Y_i as r_i defined in Eq. (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 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.

5. Model Learning

5.1 Decision Making Policy

Stepwise learning occurs online as reward sampling goes on, so that graph inference updates itself based on Eq. (23) whenever new information on reward list $\boldsymbol{r}$ is available. In order for online learning to be effective in data prediction, rules are necessary to regulate which action to take at every step. Concretely, given model inference $\hat{\boldsymbol{y}}^{(t)}$, $\boldsymbol{\sigma}_{\hat{\boldsymbol{y}}}^{(t)}$ by Formula (23), (24) 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 Fig. 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 $\boldsymbol{\sigma}_{\hat{\boldsymbol{y}}}^{(t)}$.

To study the trade-off between exploration and exploitation, this paper covers three suites of decision making policy as discussed below.

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. Commonly used acquisition functions include probability of improvement (PI), expected improvement (EI) and upper confidence bound (GP-UCB) [14], [15]. 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 of three acquisition functions used in this paper.

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

where r_{best} is current best (largest) sample reward across all the nodes so far and $\xi = 0.01$ as constant bias. Φ is the standard Gaussian cumulative density.

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

where $z = (\hat{\boldsymbol{y}} - r_{best} - \xi)/\boldsymbol{\sigma}_{\hat{\boldsymbol{y}}}$ and ϕ is the standard Gaussian probability density.

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

where x is total number of current observations and $\beta(x) = 2log(dx^2\pi^2/6\delta)$ [14]. d is dimension of node indices and $\delta = 0.9$. GP-UCB is considered a common option for Gaussian process regression and this paper tentatively takes it for comparative experiments on GMRF models.

5.1.2 Epsilon Greedy

Epsilon greedy (EPS) is a classical and naive exploration-exploitation trade-off heuristic that allocates a probability $1 - \epsilon$ for exploitation behavior and ϵ for else. In this paper $\epsilon = 0.2$.

$$\pi \leftarrow \underset{i}{\text{argmax}}\, \hat{\boldsymbol{y}} \text{ with probability } 1 - \epsilon \text{ else random } i \in D^d$$

5.1.3 Thompson Sampling

Thompson Sampling (TS) tends to trust model inference but from every node y_i takes a random sample based on its posterior $\mathcal{N}(\hat{y}_i, \sigma_i)$.

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

This paper also designs a more sophisticated variation of Thompson sampling dedicated to the proposed graphical model. Details are discussed along with experiment setup in the next section.

6. Experiments and Evaluation

6.1 Hyperparameter Tuning

The four hyperparameters α, γ_y, γ_0, γ 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 α 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 α. Based on Formula (11) it is clear that γ_0 and γ serve as smoothing parameters for interpolated means $\tilde{\mu}_i$ so we want γ_0 to be much smaller than γ to avoid over-scaling μ_i. Since magnitude of γ is responsible of connection strength $\tilde{\gamma}_i$ we experimented on γ 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 Eq. (9) so it controls how much confidence hidden nodes gain from observations. Increasing γ 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 γ_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 γ_y cripples inter-node belief propagation, which is to be avoided when adjacent nodes are expected to be correlated arms; oppositely too small γ_y grants too much correlation to the Markov random field and this makes the graphical model inclined to make incorrect inference on the truly

Algorithm 1 Synthetic Function Learning

1: Initialize hyperparameters γ, γ_y, γ_0, α
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**

optimal arm whose neighbors are of low rewards. In extreme cases, huge γ_y completely prevents a target node from influencing neighborhood and tiny γ_y allows confidence to flow freely due to too much correlation.

6.2 Experiment 1 - Learning Synthetic Functions

Wrapping up the learning model and decision making policy, we present a complete online algorithm as Algorithm 1 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 is accurate enough in approximating the environment by predicting the optimal index to achieve regret convergence.

The four hyperparameters mentioned in Section 2 require initialization before any learning takes place. Throughout this paper, $\alpha = 0.001$, $\gamma_y = 0.01$, $\gamma = 0.02d$, $\gamma_0 = 0.01\gamma$, d representing index dimension, i.e., dimension of environment as previously stated. Algorithm 1 is repeated on three different dimensions of environment so that $d = 1, 2, 3$. For each case, node index $i \in D^d$. Graph node priors 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-approaches mentioned in Section 5. For evaluation purpose only, Algorithm 1 also keeps track of regret in every iteration and computes the cumulative average from iteration 1 to t as $\bar{R}_t$ (Lines 3, 8, 9). Ideally average regret close to 0 is expected after enough rounds of iterations passed. Experiment 1 is practically designed to compare performance difference among policies in terms of minimum average regret and how fast that value is achieved. The synthesized function f used as environment varies by d and settings are respectively given below.

1D Environment

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

Environment $f(i)$ is the sum of three Gaussian density functions with separate means and variances. Sampling noise $\epsilon \sim \mathcal{N}(0, 0.025)$ and is applied to all types of environment in this paper 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 μ_1. Similar composition goes with environment in 2D and 3D cases, where multivariate Gaussian density is administered.

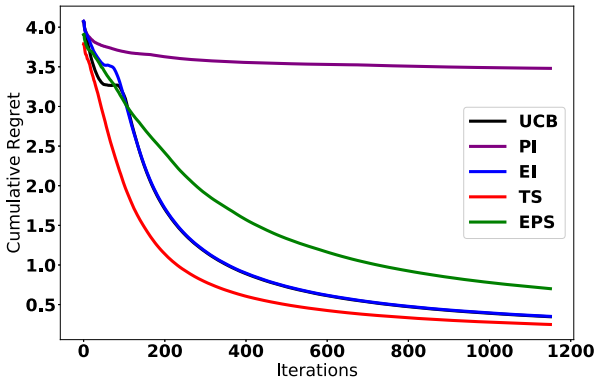

Fig. 4 1D Test - Synthesized Environment.

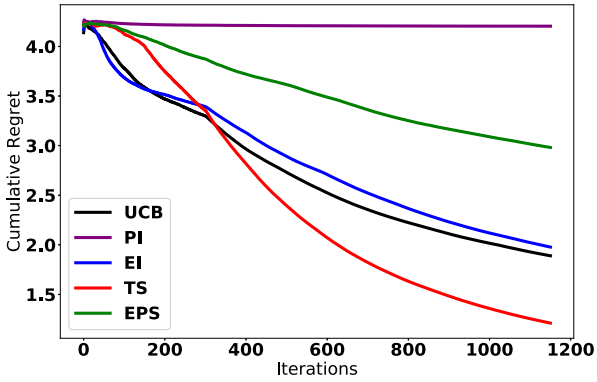

Fig. 5 2D Test - Synthesized Environment.

2D Environment

$$f(i) = \phi(i; \boldsymbol{\mu}_1, \Sigma_1) + \phi(i; \boldsymbol{\mu}_2, \Sigma_2) + \phi(i; \boldsymbol{\mu}_3, \Sigma_3) + \epsilon$$
$$\boldsymbol{\mu}_1 = \langle -3, 3 \rangle, \boldsymbol{\mu}_2 = \langle 0, 0 \rangle, \boldsymbol{\mu}_3 = \langle 3, -3 \rangle$$
$$\Sigma_1 = \begin{pmatrix} 0.025 & 0 \\ 0 & 0.025 \end{pmatrix}, \Sigma_2 = \begin{pmatrix} 2.25 & 0 \\ 0 & 2.25 \end{pmatrix}, \Sigma_3 = \begin{pmatrix} 0.25 & 0 \\ 0 & 0.25 \end{pmatrix}$$

3D Environment

$$f(i) = 3.0 \times [\phi(i; \boldsymbol{\mu}_1, \Sigma_1) + \phi(i; \boldsymbol{\mu}_2, \Sigma_2) + \phi(i; \boldsymbol{\mu}_3, \Sigma_3)] + \epsilon$$
$$\boldsymbol{\mu}_1 = \langle 3, 3, 3 \rangle, \boldsymbol{\mu}_2 = \langle -2, -2, -2 \rangle, \boldsymbol{\mu}_3 = \langle 0, 0, 0 \rangle$$
$$\Sigma_1 = \begin{pmatrix} 3 & 0 & 0 \\ 0 & 3 & 0 \\ 0 & 0 & 3 \end{pmatrix}, \Sigma_2 = \begin{pmatrix} 0.5 & 0 & 0 \\ 0 & 0.5 & 0 \\ 0 & 0 & 0.5 \end{pmatrix}, \Sigma_3 = \begin{pmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 2 \end{pmatrix}$$

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 1,000. For $d = 1$ the grid increment is 0.01. For $d = 2, 3$ node indices are from $X \times X$ and $X \times X \times X$ with larger grid increments. In Experiment 1, $T = 1{,}150$ iterations are tested for $d = 1, 2$ and 1,075 iterations for $d = 3$.

Figures 4, **5** and **6** demonstrate cumulative regrets from experiments in all three cases of distinct d. For each case, 30 repeated trials are conducted and averaged for plotting. 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 $d = 3$.

6.3 Experiment 2 - Learning Recommendation Scheduling

Experiment 1 shows how a synthesized function can be learnt

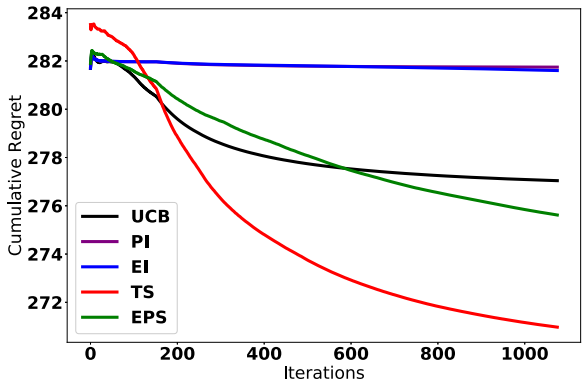

Fig. 6 3D Test - Synthesized Environment.

Algorithm 2 Learning Recommendation Scheduling

1: Initialize hyperparameters γ, γ_y, γ_0, α
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 χ ▷ 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**

from scratch through reward sampling. However, a weight list is required for decision making instead of a single index in real application where a full collection of candidate items are pending for priority assignment. So instead of an optimal index we are more interested in knowing the CTR posteriors of all the candidate items. 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. Experiment 2 presents Algorithm 2 as a modified version of Algorithm 1. According to setup of Experiment 2, $N = 100{,}000$ in analogy to 100,000 ads as available budget. The major difference in Algorithm 2 from Algorithm 1 is that it requires weight list w_i (Line 6) from policy. We expect Algorithm 2 to deliver $w_i N$ ads on item y_i given n candidate items represented by n graph nodes.

The way of constructing weight list w_i differs depending on which type of policy ($ACQ/EPS/TS$) is used, as defined in each case below.

Weights from Acquisition Function

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

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

Weights from Epsilon Greedy

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

Weights from Modified Thompson Sampling

As graph posterior $\hat{\boldsymbol{y}}, \boldsymbol{\sigma}_{\hat{\boldsymbol{y}}}$ in fact delivers 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 $p(\hat{y}_i >= y_{j \neq i}) = \prod_{j \neq i} p(\hat{y}_i >= y_j)$. Let Φ_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.

$$p(\hat{y}_i >= y_{j \neq i}) = \prod_{j \neq i} \Phi_j(\hat{y}_i) \tag{25}$$

Formula (25) 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.

$$E[p(y_i >= y_{j \neq i})] = \int_{-\infty}^{\infty} p(\hat{y}_i >= y_{j \neq i})\, d\hat{y}_i \tag{26}$$

$$= \int_{-\infty}^{\infty} \prod_{j \neq i} \Phi_j(\hat{y}_i)\, d\hat{y}_i \tag{27}$$

Therefore weights w_i are a proportion list by expectation in Eq. (26).

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

The CTR test bench χ 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 6.2. Hyperparameters are initialized with the same values in Algorithm 1. 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.

In addition to weight assignment Algorithm 2 also differs from Algorithm 1 in sampling procedure (Lines 9–12), which updates reward pattern from every node completely offline. This in practice indicates that the process of updating the model incurs no extra cost once N ads get distributed because collecting sample rewards is the only operation that requests data from real world environment (Line 7). Similar to Experiment 1, environment in different dimensions are respectively tested and evaluation is averaged across 30 trials. **Figures 7**, **8** and **9** display average missing clicks as regrets under different policies. Since Algorithm 2 performs sampling across all graph nodes, it leads to much faster regret convergence in $T = 100$ 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. Thompson sampling based on posterior expectation stands out as optimal policy in CTR prediction.

6.4 Further Discussion on Experiments

In Experiment 2, computing the integral in Eq. (26) can poten-

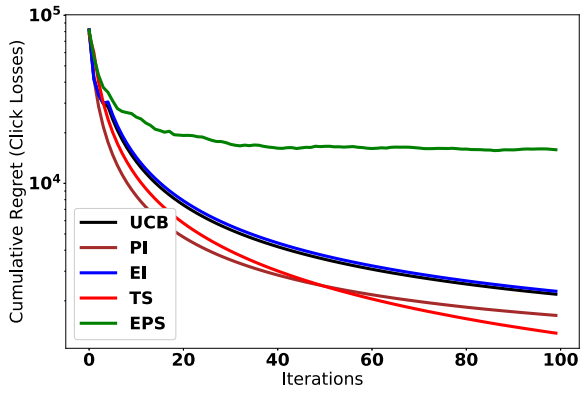

Fig. 7 1D Test - Test Bench Environment.

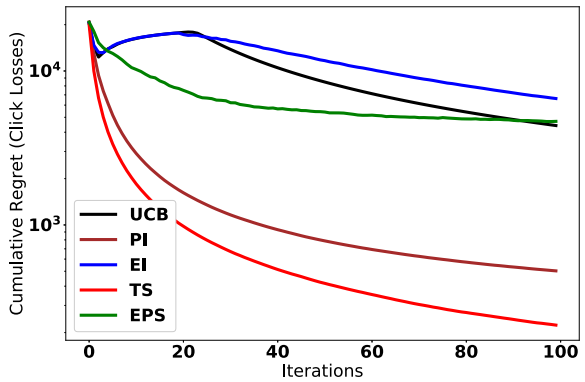

Fig. 8 2D Test - Test Bench Environment.

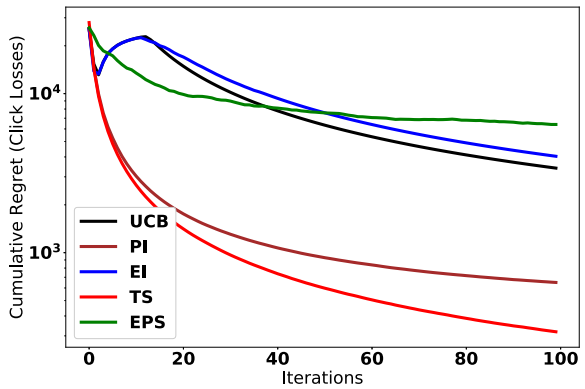

Fig. 9 3D Test - Test Bench Environment.

tially lead to numerically unstable issues due to a small distribution range $[0, 1]$ of posteriors of y_i. This boundary is explained by CTR reward observations r_i (Line 8 in Algorithm 2) 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

$$r_i = logit(\hat{y}_i) = \ln\left[\frac{\hat{y}_i}{1 - \hat{y}_i}\right]$$

The evaluations in Figs. 7, 8 and 9 are all conducted using the mapped r_i to update the graphical model.

While both Algorithm 1 and 2 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 2 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.

It is worth noting that 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. We rely on the test bench for absolute possession of the ground truth f so that cumulated regret can be computed bias free. Estimating ground truth in real-world application remains an open problem. Two common approaches of simulating "golden standard" CTRs include simply averaging [9] CTRs of an ad i as the function value $f(i)$ and excluding rarely clicked items [18] 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 paper we focus on verification of model predictivity and consider data wrangling in application dependent scenarios as future work.

7. Relate Works

The overall purpose of designing the proposed solution in this paper is to relieve the cubic complexity $O(m^3)$ nonparametric Gaussian process (GP) suffers from [11], [19] given 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 [5] and parallelizable experiment design [4] 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.

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 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 [7], using hierarchical context feature representation [21] for efficient exploration and combining solutions to the MAB problem with similarity based ranking algorithm to select subsets [17] within item-context space.

8. Concluding Remarks

This paper proposes an abstract model with GMRF property for learning from sparse data samples and predicting distribution of reward functions at discrete indices. Predictions are used to optimize the multi-armed bandit problem at best achievable cost under 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 bandits.

References

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

[2] Bubeck, S., Munos, R. and Stoltz, G.: Pure exploration in multi-armed bandits problems, *International Conference on Algorithmic Learning Theory*, pp.23–37, Springer (2009).

[3] Chapelle, O. and Li, L.: An Empirical Evaluation of Thompson Sampling, *Proc. 24th International Conference on Neural Information Processing Systems, NIPS'11*, pp.2249–2257, Curran Associates Inc. (2011).

[4] Desautels, T., Krause, A. and Burdick, J.W.: Parallelizing exploration-exploitation tradeoffs in gaussian process bandit optimization, *The Journal of Machine Learning Research*, Vol.15, No.1, pp.3873–3923 (2014).

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

[6] Honda, J. and Takemura, A.: An asymptotically optimal policy for finite support models in the multiarmed bandit problem, *Machine Learning*, Vol.85, No.3, pp.361–391 (2011).

[7] Li, L., Chu, W., Langford, J. and Schapire, R.E.: A contextual-bandit approach to personalized news article recommendation, *Proc. 19th International Conference on World Wide Web*, pp.661–670, ACM (2010).

[8] Maes, F., Wehenkel, L. and Ernst, D.: Learning to play K-armed bandit problems, *Proc. 4th International Conference on Agents and Artificial Intelligence* (*ICAART 2012*) (2012).

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

[10] Nguyen, T.V., Karatzoglou, A. and Baltrunas, L.: Gaussian process factorization machines for context-aware recommendations, *Proc. 37th International ACM SIGIR Conference on Research & Development in Information Retrieval*, pp.63–72, ACM (2014).

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

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

[13] Schreiter, J., Nguyen-Tuong, D., Eberts, M., Bischoff, B., Markert, H. and Toussaint, M.: Safe Exploration for Active Learning with Gaussian Processes, *Proc. 2015th European Conference on Machine Learning and Knowledge Discovery in Databases - Volume Part III, ECMLPKDD'15*, pp.133–149, Springer (2015).

[14] Srinivas, N., Krause, A., Kakade, S. and Seeger, M.: Gaussian Process Optimization in the Bandit Setting: No Regret and Experimental Design, *Proc. 27th International Conference on International Conference on Machine Learning, ICML'10*, pp.1015–1022, Omnipress (2010).

[15] Srinivas, N., Krause, A., Kakade, S.M. and Seeger, M.W.: Information-theoretic regret bounds for gaussian process optimization in the bandit setting, *IEEE Trans. Information Theory*, Vol.58, No.5, pp.3250–3265 (2012).

[16] Sui, Y., Gotovos, A., Burdick, J.W. and Krause, A.: Safe Exploration for Optimization with Gaussian Processes, *Proc. 32nd International Conference on International Conference on Machine Learning - Volume 37, ICML'15*, pp.997–1005, JMLR.org (2015).

[17] Vanchinathan, H.P., Nikolic, I., De Bona, F. and Krause, A.: Explore-exploit in top-n recommender systems via gaussian processes, *Proc. 8th ACM Conference on Recommender Systems*, pp.225–232, ACM (2014).

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

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

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

[21] Yue, Y., Hong, S.A. and Guestrin, C.: Hierarchical Exploration for Accelerating Contextual Bandits, *Proc. 29th International Coference on International Conference on Machine Learning, ICML'12*, pp.979–986, Omnipress (2012).

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

[23] Zhao, C., Watanabe, K., Yang, B. and Hirate, Y.: Fast Converging Multi-armed Bandit Optimization Using Probabilistic Graphical Model, *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pp.115–127, Springer (2018).

Appendix

A.1 Proof of the Model Likelihood Factor

Section 3 claims that the negative log likelihood (15) is equivalent to matrix multiplication in Eq. (17). $\boldsymbol{A}$, $\boldsymbol{B}$ is copied here as a side note. Proof is given below.

$$\boldsymbol{A} = \begin{pmatrix} \gamma_y k_1 + \tilde{\gamma}_1 & -\gamma_y & \cdots & -\gamma_y & \cdots & & \\ -\gamma_y & \gamma_y k_2 + \tilde{\gamma}_2 & \cdots & \cdots & -\gamma_y & & \\ \vdots & \vdots & \ddots & & & & \\ -\gamma_y & \vdots & & \ddots & & & \\ & -\gamma_y & & & \ddots & & \\ & & \cdots & & & & \gamma_y k_n + \tilde{\gamma}_n \end{pmatrix}$$

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

Given $E(\boldsymbol{y}) = \sum_i \left\{\tilde{\gamma}(y_i - \tilde{\mu}_i)^2\right\} + \sum_{i,j\in E} \left\{\gamma_y (y_i - y_j)^2\right\}$, we want to prove $E'(\boldsymbol{y})$ below is only an alternative expression to $E(\boldsymbol{y})$.

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

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

Proof. $E(\boldsymbol{y}) = E'(\boldsymbol{y})$ 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 $\Rightarrow$

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

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

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

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

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

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

Recall that Section 3 mentions that k_x is 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

$\sum_{i=1}^n y_i^2 k_i = \sum_{i,j\in E} y_i^2 + \sum_{i,j\in E} y_j^2$

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

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

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

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

A.2 Proof of the Closed-form Solution

For this proof, we assume $\boldsymbol{A}$ is positive definite, a conclusion that is proved in Appendix A.3, so that $\boldsymbol{A}$ 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.

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

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

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}}) \\
&\quad - \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}} \\
&\quad - \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}}) \\
&\quad + \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}}) \\
&\quad + \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}}) \\
&\quad + \tilde{\boldsymbol{\mu}}^\top (\boldsymbol{B} - \boldsymbol{B}\boldsymbol{A}^{-1}\boldsymbol{B})\tilde{\boldsymbol{\mu}}
\end{aligned}$$

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

A.3 Proof on Positive Definiteness

Appendix A.2 assumes positive definiteness on $\boldsymbol{A}$, a necessary condition for $\boldsymbol{A}$ to possess unique Cholesky decomposition in the form of $\boldsymbol{D}^\top \boldsymbol{D}$.

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.*

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

The objective of this section is to prove that $\boldsymbol{A}$ is positive definite. Let v be a size-n non-zero real vector. Based on how $\boldsymbol{A}$ is constructed in Eq. (16), $\boldsymbol{v}^\top \boldsymbol{A}\boldsymbol{v}$ can be expanded as follows.

Proof. $\boldsymbol{A}$ is positive definite.

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

As discussed in Section 2.2.2 $\tilde{\gamma}_i = m_i\gamma + \gamma_0$. m_i is the sample size of target node y_i. As hyperparameters γ and γ_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 to Appendix A.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 $\sum_{i=1}^n v_i^2 \gamma_y k_i = \sum_{i,j\in E} v_i^2 \gamma_y + \sum_{i,j\in E} v_j^2 \gamma_y$.

$$\sum_{i=1}^n v_i^2 \gamma_y k_i - 2\sum_{i,j\in E} \gamma_y v_i v_j = \sum_{i,j\in E} v_i^2 \gamma_y + \sum_{i,j\in E} v_j^2 \gamma_y - 2\sum_{i,j\in E} \gamma_y v_i v_j$$

$$\Rightarrow \sum_{i=1}^n v_i^2 \gamma_y k_i - 2\sum_{i,j\in E} \gamma_y v_i v_j = \sum_{i,j\in E} \gamma_y (v_i - v_j)^2$$

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

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

Chen Zhao was born in 1988. He is currectly a Ph.D. candidate Deparment of System and Information Engeering, University of Tsukuba. He received his M.S. degree from Department of Computer Engeering, Iowa State University in 2014. His research lies in the field of natural language processing, statistical learning and information retrieval. He was working as a part-time machine learning reseacher in Rakuten Institute of Technology, Tokyo in 2017.

Bin Yang received his Ph.D. in 2013 from Graduate School of Information Science and Technology, the University of Tokyo. He is currently working at Hitachi R&D Group as a senior researcher. His research interests are in Machine Learning, Data Privacy, Bayes Statistics and Data Mining.

Yu Hirate was born in 1980. He received his Dr.Eng. degree from Department of Computer Science in Waseda University in 2008 and was an assistant professor of the Media Network Center, Waseda University during 2006–2009. He joined Rakuten Institute of Technology in 2009 and is now the principle scientist and senior manager.

(Editor in Charge: *Suzuki Yu*)