Aalto University
School of Science
Master's Programme in Computer, Communication and Information Sciences

An-Dan Nguyen

Bayesian Optimization for Partially Overlapping Covariate Data Sources

Master's Thesis
Espoo, January 24, 2022

Supervisor: Professor Samuel Kaski, Aalto University
Advisors: Jussi Gillberg M.Sc. (Tech.), Linh Nguyen M.Sc. (Tech.)

Aalto University
School of Science
Master's Programme in Computer, Communication and Information Sciences

ABSTRACT OF MASTER'S THESIS

Author: An-Dan Nguyen

Title:
Bayesian Optimization for Partially Overlapping Covariate Data Sources

Date: January 24, 2022 **Pages:** 57

Major: Computer Science **Code:** SCI3042

Supervisor: Professor Samuel Kaski

Advisors: Jussi Gillberg M.Sc. (Tech.), Linh Nguyen M.Sc. (Tech.)

One problem in the real-world industrial process is how to utilize diverse information on best practices through different data sources. It becomes more complicated when those best practices are different, but not entirely, from each other. The goal is to find the optimal best practices from those diverse and somewhat different data.

That problem has been formulated into finding the optimal parameter settings in diverse, partially overlapping covariate data sources. First, the data from different sources are stacked row-wise to form a master data set with missing data. Then, Bayesian Optimization with Missing Inputs is employed to find the optimal experiment's parameter settings.

Different methods of modeling the missing data set are tested, such as Bayesian Non-negative Matrix Factorization (BNMF) and Bayesian Probabilistic Matrix Factorization (BPMF). Both provide a quality representation of the missing data, allowing the Bayesian Optimization algorithm to work. The BPMF-based methods have significantly better performances than the BNMF-based methods. However, BNMF-based methods are helpful in some specific cases due to the structure of the missing data set.

Multi-armed Bandit Algorithms are used to tackle the problem of a parameter settings budget constraint in each iteration. The ϵ-greedy and UCB1 have been tested. The ϵ-greedy can occasionally give better results because of its randomness. In contrast, The UCB1 consistently improves its performance through each iteration.

This work proposes a framework to utilize the information from partially overlapping data sources to find the parameter settings that yield a maximum return. This work benefits a wide range of real-world industrial production processes and opens exciting research directions.

Keywords: bayesian optimizaion, multi-source, partial overlapping, missing data, multi-armed bandits, matrix factorization, acquisition functions

Language: English

Acknowledgements

I wish to deeply thank my advisors, especially Jussi Gillberg, who has provided me with a chance to work on this exciting topic and countless invaluable suggestions and comments. I also want to thank Professor Samuel Kaski, whose critical feedback has guided me to complete the thesis. I want to express my sincere gratitude to my advisor Linh Nguyen who has helped me in crucial moments throughout my master's degree from the start to the end. Finally, this master thesis could not be completed without the help of friends in Finland, Singapore, Vietnam, and my dear family in Vietnam, who remind me to keep focused and finish what I have started.

Espoo, January 24, 2022

An-Dan Nguyen

Abbreviations and Acronyms

BO	Bayesian Optimization
BNMF	Bayesian Non-negative Matrix Factorization
BPMF	Bayesian Probabilistic Matrix Factorization
BOMI	Bayesian Optimization with Missing Inputs
GP	Gaussian Process
GP-UCB	Gaussian Process Upper Confidence Bound
UCB1	Multi-arm Bandit Upper Confidence Bound
UCB-MI	Upper Confidence Bound with Missing Inputs

Contents

Chapter 1

Introduction

1.1 Motivation

In several industries, the optimisation of industrial manufacturing processes, such as pharmaceutical production and carbon fiber manufacturing, is critical for successful commercial production. While the outcome of a production process depends on numerous parameters, acquiring representative data to quantify parameters' impacts and interactions for optimisation purposes is challenging because different manufacturers modify (skip, replace) some of the steps in the standard processes to fit their needs and as a result, the data sets relating to the same target process recorded by different manufacturers do not necessarily even contain values of the same parameters.

An almost similar problem arises when the objective is to combine several "not so different" industrial process into a new general process to improve yield. In such a case, one does not have a data set containing values of all process parameters to optimize. Instead, one can only access data sets from the similar industrial procedures where at best the parameters recorded for the different processes overlap.

Importantly, complex real-world manufacturing processes are often expensive to run in terms of time, money, or both. Furthemore, modifying the value of each parameter comes at a cost. Due to these cost factors, one cannot experiment with all process parameters simultaneously (as in a computational experiment, for example) but one rather can vary the values of only a small subset at a time. For example, running one industrial process end-to-end to produce a complete product can last several weeks while requiring an expensive qualified laboratory environment and modifying the value of each parameter may take days of work. These cost factors require taking into account the budget constraints when performing experiments to

identify optimal parameter settings. The budget constraint can relate to the maximum number of experiment iterations or the number of parameters whose values can be changed in each experiment.

This thesis develops a novel framework to address the needs mentioned above.

1.2 Problem statement

To formulate the problem, let's first define:

- The outcome of the industrial process is modelled as function f.
- The values of the n parameters of the industrial process f are collected in an n-dimensional vector x referred to as the setting. X denotes the distribution of settings. x is drawn from X.
- $f(x)$ or y is the yield of a specific setting x of the industrial process f.

For generality, a general industrial f_{gen} is assumed to be a black-box function. As varying the values of the parameters of an industrial process is often costly, it is reasonable to assume that f_{gen} is expensive to evaluate.

To develop a solution to address the needs presented in Section 1.1, Bayesian Optimization (BO) is used as a starting point. BO is an optimization method where the function f is a black-box function. BO assumes that the analytical form of f is not known. Thus, the derivative of f cannot be evaluated analytically. Furthermore, the function f is expected to be very expensive to evaluate, and only a limited number of values of $f(x)$ (in this case, the yields) can be obtained. Hence, grid search or using numeric gradient estimate to find the optimal value [19] is not practical. Bayesian Optimization techniques are hence well suited the problem studied in this thesis. By modeling the function f using the data $(x_n, f(x_n))$ and a surrogate model (often a Gaussian Process), optimal values of f can be sought in a computationally feasible manner.

Conventionally, information for the full set of covariates in X from which x is drawn is available, and Bayesian Optimization has been proven quite beneficial in such cases [50][72]. However, for certain industrial processes, data about the settings of similar industrial processes contain measurements from different but partially overlapping subsets of the covariates from X covariate sets. Assuming that the union of covariates from those data sources is the exhausting set of covariates of X, and there are no missing in the yields $(y = f(x))$, the problem of applying Bayesian Optimization where the data

of the covariates are presented only in some data sources is formalized. The problem is illustrated by the Figure 1.1 below. X can be seen as a general process in this picture. And X_1, X_2, X_3 are the modified versions of the general procedure where missing covariates are the skipped or modified steps, mentioned in Section 1.1

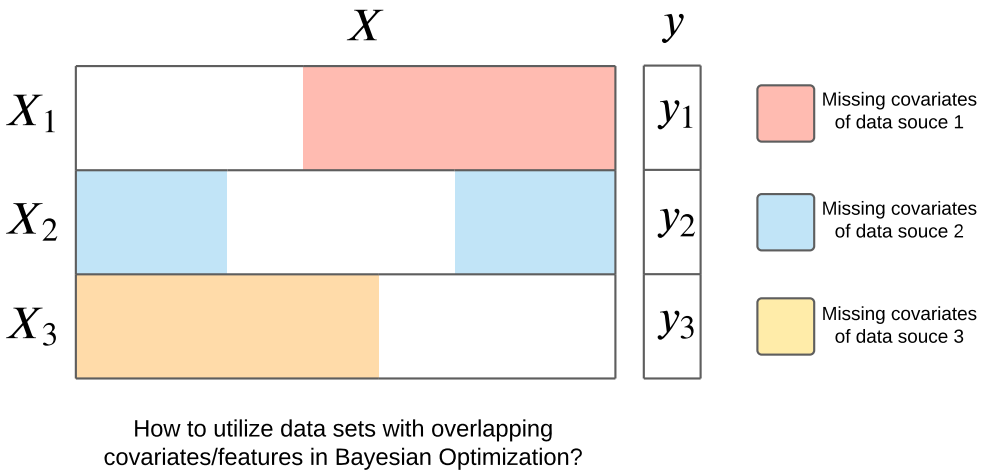


Figure 1.1: Summary of Bayesian Optimization in partially overlapping covariates data sources

1.3 Structure of the Thesis

The thesis consists of eight chapters. Except for Chapter 1, the rest of the work is structured as follows: Chapter 2 presents the survey of the literature on Bayesian Optimization, missing data, missing data imputation methods, and works related to our study. Next, Chapter 3 briefly presents the background knowledge related to the to-be-developed methods. In Chapter 4, the problem is formally defined, and the details of the novel method to solve the problem are presented. Then, in Chapter 5, the setup of the experiments to verify the methods is laid out. The data used in these experiments is presented in this chapter. The evaluation methods and metrics are also mentioned in this chapter. Subsequently, in Chapter 6, the results of the experiments and how well they are compared to each other are carefully explained. Moreover, in Chapter 7 a discussion of the methods and the findings can be found. Lastly, conclusions are drawn in Chapter 8.

Chapter 2

Related Work

Bayesian Optimization (BO) was first introduced by Kushner [36], Zhilinskas [81], Mockus [47] [45] and popularized by Efficient Global Optimization, which is the work of John et al. [29]. BO is a vast area of research with different topics such as multi-objective optimization [46][33][31], multi-fidelity optimization [27][65], and convergence rate [10][8][9][7] to name a few.

Bayesian Optimization has also been adopted in various application domains such as agriculture [50][58][69][71] and industrial production [79][11][43] [49][12]. With the rise in popularity of the deep neural networks, BO is also a handy tool in their hyper-parameters optimization [32].

Few works have applied BO in the setup similar to what has been described in Chapter 1. Nevertheless, the formulation in Chapter 4 opens a new approach to address the problem, i.e., formulating the problem in terms of BO for missing data set with constraint budget. The literature that connects to that problem formulation is listed below.

Missing data research is as extensive as Bayesian Optimization research. Buuren's book [73] is comprehensive coverage of imputation techniques for missing data. However, according to Luong et al. [42], the imputation techniques are not as effective compared to likelihood-based approaches when it comes to the context of applying BO in the missing data set. The results align with a comment in Buuren's book related to likelihood-based approaches. There are several notable likelihood-based approaches in the literature in dealing with missing data in general, such as [14] [39] and others in the context of Kernel Methods and Gaussian Processes [63] [62]. In this work, based on the suggestion in Luong et al. above, the techniques of Bayesian Matrix Factorization to deal with the data missingness are employed. Chapter 3 introduces those techniques in details.

Another line of work in Bayesian Optimization dealing with the constraint budget related to the methods presented in this thesis can be mentioned in

articles [64][37][41]. These methods are not suitable in this situation since they require a fully observed covariate x to evaluate the objective function $f(x)$. This requirement is not satisfied due to the budget constraint in choosing the covariates in each evaluation. The recently published work of Hayashi et al. [24] is the closest to the setup of this thesis where the experiment's parameters can be changed is fixed, and others are unobserved or cannot be controlled. The difference is that we can choose from the set of all parameters but not all of them simultaneously due to budget constraints. Meanwhile, they are only allowed to choose the parameters from a fixed subset of all parameters.

The next chapter presents the technical details of the to-be-implemented methods to solve the problem.

Chapter 3

Background

This section introduces the background of the main methods introduced in the later chapters. These are Bayesian Optimization, Bayesian Matrix Factorization methods, the Bayesian Optimization with Missing Inputs, and the Multi-armed Bandit Problem.

3.1 Bayesian Optimization

Bayesian Optimization is a black-box optimization methods that attempt to solve the following problem:

$$x^* = \arg\max_{x \in A} f(x)$$

where f is called the objective function and A is called a feasible set. The objective function and feasible set often have these properties:

- The input x is in $\mathbb{R}^d$. Traditionally, d is typically not too large ($d \leq 20$). However, later advancements helped Bayesian Optimization deal with high-dimensional data sets.
- The simple set A is a feasible set which membership is easily assessed. Normally $x \in A$ is defined as $\{x \in \mathbb{R}^d : a_i \leq x \leq b_i\}$ or $\{x \in \mathbb{R}^d : \sum_i x_i = 1\}$.
- The objective function f is continuous to model using Gaussian process regression.
- The objective function f is expensive to evaluate both in terms of the time it takes to evaluate f (e.g., hours in computer experiments, days in industrial processes, or months in agriculture processes) or the

economic/opportunity cost to evaluate f is high (e.g., industrial and agriculture experiments take much money to run).

- Since f is a black-box function, the form of f is not known to apply techniques that leverage the knowledge for optimization.
- When evaluating f, only the value of $f(x)$ is known but not its first or second derivative. Therefore, the methods such as gradient descent, Newton's method, or quasi-Newton methods cannot be applied. This property is called derivative-free.
- Assuming no noise when observing $f(x)$.
- Finding the global optimum, rather than the local optimum, is the focus.

In short, according to Frazier [19], Bayesian Optimization can be seen as a black-box derivative-free global optimization. The tutorial in Bayesian Optimization of Frazier is also comprehensive material for further reference. Next, the mechanism of Bayesian Optimization and its components are described.

3.1.1 Overview

There are two main components in Bayesian Optimization:

- The first component is a probabilistic statistical to model the black-box objective function. This component is also called a surrogate model.
- The second component is an acquisition function to suggest the following sample to evaluate.

The steps to perform Bayesian Optimization are presented in the pseudo-code 1. The black-box objective function is f and the budget (number of experiments) to evaluate f is N.

Next, two essential components of Bayesian Optimization are presented. They are surrogate models and acquisition functions.

3.1.2 Surrogate Models

There are two choices for probabilistic surrogate models of Bayesian Optimization. They are Gaussian Processes [55] and Random Forests [4]. Traditionally, Gaussian Processes have been chosen as the off-the-shelf surrogate model's Bayesian Optimization because of its expressiveness, closed-form analytical formula, and built-in uncertainty estimation. However, they scale

Algorithm 1 Pseudo-code for Bayesian Optimization

Evaluate k random points with f to form the basis data set $\{x_i, f(x_i))_{i=0}^{n=k}\}$.
Model the data $\{x_i, f(x_i))_{i=0}^{n=k}\}$ with a probabilistic statistical model.
Set $i = k$.
while $i \leq N$ **do**
Update the posterior probability distribution of f using all the data.
Compute the acquisition function using the updated posterior distribution.
Choose the x_{next} by select the maximizer of the acquisition function over x.
Evaluate $f(x_{next})$. Then add the new point $(x_{next}, f(x_{next})$ into the current data set.
Increase i.
end while

poorly regarding the number of dimensions and samples in the data set. Therefore, there have been a number of researches seek to alleviate those problems [22][76][77][20][30].

Random Forests are another choice for a surrogate model in Bayesian Optimization. If Gaussian Processes are well-suited in low dimensional, numerical configuration spaces problems [17], Random Forests are useful in high-dimensional, conditional configuration spaces and partly discrete ones [28][38]. Moreover, the tree-based method reduces the training time compared to Gaussian Processes when dealing with a large data set. Since it is well known that Gaussian Processes running time grows cubically in terms of training examples and parameters. Meanwhile, it only grows linearly in tree-based methods. There is also research that uses another tree-based method which is the gradient-boosted tree, to serve as a surrogate model [74].

Gaussian Processes are chosen as surrogate models in this thesis since the data set is low dimensional with numerical parameters. The basics of Gaussian Process Regression are briefly presented in the next section. A complete and comprehensive introduction of Gaussian Process Regression and Gaussian Process can be found in the book of Williams and Rasmussen [78].

3.1.2.1 Gaussian Process Regression

A Gaussian Process can be seen as a probabilistic distribution over continuous function. Suppose there are the mean function $m(\mathbf{x})$ and the covariance

function ($k(\mathbf{x}, \mathbf{x}')$) of a real process $f(\mathbf{x})$:

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

then the Gaussian Process is defined as:

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

Suppose there exists the data set of n elements $\mathcal{D} = \{(\mathbf{x_i}, y_i))\}_{i=1}^{n}$ where $y_i = f(\mathbf{x_i}) + \epsilon_i$ for $i = 1, 2, ..., n$ and $\epsilon_i \sim \mathcal{N}(0, \sigma_\epsilon^2)$, the predictive equation of the test point $\mathbf{x}_*$ of Gaussian Process Regression in this case of noisy observations is:

$$f(\mathbf{x}_*)|\mathcal{D} \sim \mathcal{N}(\mu(\mathbf{x}_*), \sigma^2(\mathbf{x}_*)) \tag{3.1}$$

with $\mu(\mathbf{x}_*$, and $\sigma^2(\mathbf{x}_*)$ is the realization of the mean function $m(\mathbf{x})$ and covariance function $k(\mathbf{x}, \mathbf{x}')$ above, in the case prediction with noisy observation:

$$\mu(\mathbf{x}_*) = {\mathbf{k}_*}^T(\mathbf{K} + \sigma_\epsilon^2\mathbf{I})^{-1}\mathbf{y} \tag{3.2}$$

$$\sigma^2(\mathbf{x}_*) = k(\mathbf{x}_*, \mathbf{x}_*) - {\mathbf{k}_*}^T(\mathbf{K} + \sigma_\epsilon^2\mathbf{I})^{-1}\mathbf{k}_* \tag{3.3}$$

where:

- $\mathbf{k}_*$ is the covariance between all the inputs $\mathbf{x_i}$ with the test point $\mathbf{x}_*$, $\mathbf{k}_* = [k(\mathbf{x_i}, \mathbf{x}_*)], \forall \mathbf{x_i} \in \mathcal{D}$.
- $\mathbf{K}$ is the covariance between all inputs, $\mathbf{K} = [k(\mathbf{x_i}, \mathbf{x_j})], \forall \mathbf{x_i}, \mathbf{x_j} \in \mathcal{D}$.
- $\mathbf{I}$ is the identity matrix that has the same dimension as $\mathbf{K}$.
- $\mathbf{y} = [y_i, ..., y_n]$ is the vector of observation with noise.
- σ_ϵ^2 is the observation noise.

The details of this derivation can be found in Chapter 2 of Williams and Rasmussen's book [78].

The choice of the covariance function $k(\cdot, \cdot)$, also called the kernel, defines the behaviour of the Gaussian Process. The common choice and simple kernel is the Gaussian kernel or squared exponential kernel:

$$k(\mathbf{x}, \mathbf{x}') = \sigma^2 \exp\left(-\frac{1}{2l^2}||\mathbf{x} - \mathbf{x}'||^2\right)$$

where σ^2 is the overall variance of the function and l is the lengthscale. The other popular choice is the Matérn kernel:

$$k(\mathbf{x}, \mathbf{x}') = \sigma^2 \frac{2^{1-\nu}}{\Gamma(\nu)} \left(\frac{\sqrt{2\nu}||\mathbf{x} - \mathbf{x}'||}{l} \right)^{\nu} K_{\nu} \left(\frac{\sqrt{2\nu}||\mathbf{x} - \mathbf{x}'||}{l} \right)$$

where σ^2 is the overall variance of the function, Γ is the gamma function, K_ν is the modified Bessel function, l, and ν are the parameters of the covariance. In order to choose the hyper-parameter, such as the lengthscale l in a squared exponential kernel of the Gaussian Process, one can use the maximize log marginal likelihood as laid out in Chapter 5 of Williams and Rasmussen's book [78].

In this work, the Gaussian kernel is chosen to use in the Gaussian Processes.

By modeling the black-box function f as a Gaussian Process through the input data $\mathcal{D} = \{(\mathbf{x_i}, y_i))\}_{i=1}^n$, the behaviour of the black-box function can be inferred by drawing samples from the learned posterior probability distribution as shown in Equation 3.1. The most rewarding point can be computed using this posterior distribution, based on pre-defined criteria, to evaluate next using the acquisition functions. The acquisition functions are presented in the next section.

3.1.3 Acquisition Functions

As stated in the last section, modeling the black-box function by Gaussian Processes can be useful in determining the next point to evaluate by using the acquisition function. The acquisition functions harness the information given by the posterior distribution model by the Gaussian Processes, especially the mean function (Equation 3.2) and the covariance function (Equation 3.3). Different combinations between the mean function and the covariance function express the trade-off between exploitation (sample the point where the expected value is high, mean function) and exploration (sample the point when the uncertainty of the return value is high, covariance function).

There a wide range of acquisition functions have been developed such as Expected Improvement [29], Knowledge Gradient [18], Entropy Search [25], Predictive Entropy Search [26], GP-UCB [67] and a modified version of GP-UCB for missing inputs called UCB-MI [42]. Since this work uses UCB-MI, which uses GP-UCB, the details of GP-UCB are presented in the next section. UCB-MI will be presented later when introducing the Bayesian Optimization for Missing Inputs of Luong et al.

3.1.3.1 Gaussian Process Upper Confidence Bound (GP-UCB)

The GP-UCB acquisition function can be computed as:

$$\alpha_t^{GP-UCB}(x) = \mu_t(\mathbf{x}) + \sqrt{\beta_t}\sigma_t(\mathbf{x})$$

We have:

- μ_t and σ_t are the mean function and covariance function learned from the data by a surrogate model up to iteration t.
- β_t is the exploitation-exploration trade off factor, and is calculate by the formula:

$$\beta_t = 2\log(t^2 2\pi^2/(3\delta)) + 2d\log\left(t^2 dbr\sqrt{\log(4da/\delta)}\right)$$

where $a, b > 0$ are constant and $r > 0$, the formula of β_t is to guarantee an upper bound on the cummulative regret with probability greater than $1 - \delta$ in the input space $\mathcal{D} \subset [0, r]^d$. Details can be found in the Theorem 2 in the paper of Srnivas et al. [67].

Following the GP-UCB, at each step t of the Bayesian Optimization loop, a next point is chosen to evaluate based on the formula:

$$x_{t+1} = \arg\max_{x \in \mathcal{D}} \alpha_t^{GP-UCB}(x)$$

Next, Bayesian Matrix Factorization techniques are described, especially the Bayesian Probabilistic Matrix Factorization (BPMF) and Bayesian Non-Negative Matrix Factorization (BNMF), since they are essential components in the methods and experiments in the following chapters.

3.2 Bayesian Matrix Factorization

Matrix factorization and decomposition, such as eigendecomposition, singular value decomposition (SVD), and others, have been widely used in machine learning and data analytics to reduce the dimension of the dataset or find a low-rank approximation. However, problems arise when those methods are applied to data sets such as The Netflix Prize challenge [3] where there are a lot of missing data (users often only rate some movies). Modifications to the traditional tools are developed to overcome such troubles such as [35][21][53][66]. However, these methods either have to rely on external data, imputation methods, or ignore the missing data completely to perform the task that significantly affects the prediction and modeling power.

Salakhutdinov and Mnih developed a probabilistic framework for matrix factorization called Probabilistic Matrix Factorization (PMF) [44] which put a prior on the factorized components. The framework helps incorporate the uncertainty of unobserved values into the model naturally. Later, they developed the Bayesian Probabilistic Matrix Factorization (BPMF) [59] in order to predict the rating for new user/movie pairs. Gibbs sampling was used for approximate inference. BPMF is briefly introduced. Later, another Bayesian Matrix Factorization technique called Bayesian Non-Negative Matrix Factorization (BNMF) is described. It puts a non-negative constraint on the sign of the elements of the factorized matrices.

3.2.1 Bayesian Probabilistic Matrix Factorization

Since the framework was developed to solve The Netflix Prize challenge, the same concept of users, movies, and ratings are used as in the original paper to describe the techniques easily.

3.2.1.1 Probabilistic Matrix Factorization (PMF)

Suppose there are N users, M movies, and R_{ij} is the rating of user i for movie j. The D-dimension user-specific and movie-specific hidden feature vectors are represented by U_i and V_j, respectively. The prior distributions over $U \in \mathbb{R}^{D\times N}$, $V \in \mathbb{R}^{D\times M}$, and the conditional distribution over observed ratings $R \in \mathbb{R}^{N\times M}$ are:

$$p(U|\alpha_U) = \prod_{i=1}^{N} \mathcal{N}\left(U_i|0, \alpha_U^{-1} I\right) \tag{3.4}$$

$$p(V|\alpha_V) = \prod_{j=1}^{M} \mathcal{N}\left(V_j|0, \alpha_V^{-1} I\right) \tag{3.5}$$

$$p(R|U, V, \alpha) = \prod_{i=1}^{N}\prod_{j=1}^{M} \left[\mathcal{N}\left(R_{ij}|U_i^T V_j, \alpha^{-1}\right)\right]^{I_{ij}} \tag{3.6}$$

where:

- $\mathcal{N}(x|\mu, \alpha^{-1})$ denotes the Gaussian distributions with mean μ and precision α.
- I_{ij} is the indication which equal to 1 if the user i rated the movie j and equal to 0 otherwise.

The model is learned by maximizing the log-posterior:

$$\log p(U, V|R, \alpha, \alpha_V, \alpha_U) = \log p(R|U, V, \alpha) + \log p(U|\alpha_U) + \log p(V|\alpha_V + C$$

Maximizing the posterior distribution with respect to U and V is the same as minimizing the sum-of-squares error function with quadratic regularization terms:

$$E = \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{M} I_{ij}(R_{ij} - U_i^T V_j)^2 + \frac{\lambda_U}{2}\sum_{i=1}^{N}||U_i||_{Fro}^2 + \frac{\lambda_V}{2}\sum_{j=1}^{M}||V_j||_{Fro}^2$$

where $||\cdot||_{Fro}^2$ is the Frobenius norm, and $\lambda_U = \alpha_U/\alpha$, $\lambda_V = \alpha_V/\alpha$. A local minimum can be found by running the descent in U and V with the objective function above.

3.2.1.2 Bayesian Probabilistic Matrix Factorization (BPMF)

Using the same notation and context in the last section 3.2.1.1, the BPMF model is presented below. The likelihood term is the same as in the Equation 3.6. The prior over user and movie feature vectors are:

$$p(U|\mu_U, \Lambda_U) = \prod_{i=1}^{N}\mathcal{N}(U_i|\mu_U, \Lambda_U^{-1})$$

$$p(V|\mu_V, \Lambda_V) = \prod_{i=1}^{M}\mathcal{N}(V_i|\mu_V, \Lambda_V^{-1})$$

A Normal-Wishart priors was placed on the user and movie hyper-parameters $\Theta_U = \{\mu_U, \Lambda_U\}$ and $\Theta_V = \{\mu_V, \Lambda_V\}$:

$$\begin{aligned} p(\Theta_U|\Theta_0) &= p(\mu_U|\Lambda_U)p(\Lambda_U) \\ &= \mathcal{N}(\mu_U|\mu_0, (\beta_0\Lambda_U)^{-1})\mathcal{W}(\Lambda_U|W_0, \nu_0) \\ p(\Theta_V|\Theta_0) &= p(\mu_V|\Lambda_V)p(\Lambda_V) \\ &= \mathcal{N}(\mu_V|\mu_0, (\beta_0\Lambda_V)^{-1})\mathcal{W}(\Lambda_V|W_0, \nu_0) \end{aligned}$$

where $\mathcal{W}$ is the Wishart distribution with a $D \times D$ scale matrix W_0 and ν_0 degrees of freedom. The probability density function (pdf) of the Wishart distribution is:

$$\mathcal{W}(\Lambda|W_0, \nu_0) = \frac{1}{C}|\Lambda|^{(\nu_0 - D - 1)/2}\exp\left(\frac{1}{2}\mathtt{Tr}(W_0^{-1}\Lambda)\right)$$

where C is thr normalizing constant. The hyper-parameters μ_0, ν_0, W_0 were set in advance. The graphical models of PMF and BPMF is shown below. The predictive posterior distribution of the new rating R^*_{ij} for user i and

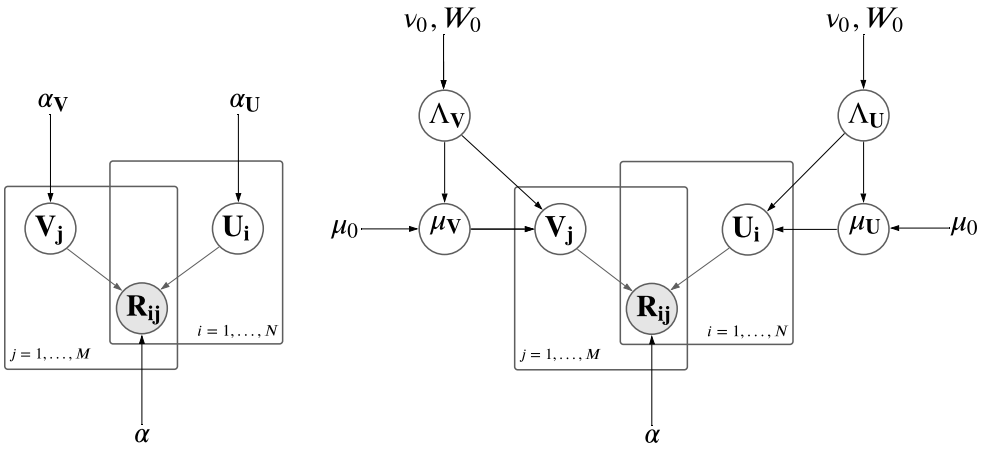


Figure 3.1: Graphical models of PMF (left) and BPMF (right). The images were reproduced from the original paper [59].

movie j can be obtained by the formula below:

$$p(R^*_{ij}|R, \Theta_0) = \iint p(R^*_{ij}|U_i, V_j) p(U, V|R, \Theta_U, \Theta_V) \\ p(\Theta_U, \Theta_V|\Theta_0) \mathrm{d}\{U, V\} \mathrm{d}\{\Theta_U, \Theta_V\}$$

Since the predictive posterior distribution is analytically intractable, a Markov Chain Monte Carlo (MCMC) based method is used to do approximate inference. The Monte Carlo approximation of the predictive posterior distribution above is:

$$p(R^*_{ij}|R, \Theta_0) \approx \frac{1}{K} \sum_{k=1}^{K} p(R^*_{ij}|U^k_i, V^k_j)$$

The author used Gibbs sampling, an MCMC algorithm, which sample the hidden variables from their conditional probability distributions over the current values of all other variables. The conditional probability distribution of user vector U_i, given the movie features, observed rating matrix R, and values of other hyper-parameters is:

$$p(U_i|R, V, \Theta_U, \alpha) = \mathcal{N}(U_i|\mu^*_i, [\Lambda^*_i]^-1) \\ \sim \prod_{j=1}^{M} \left[\mathcal{N}(R_{ij}|U^T_i V_j, \alpha^{-1})\right]^{I_{ij}} p(U_i|\mu_U, \Lambda_U)$$

where:

$$\Lambda_i^* = \Lambda_U + \alpha \sum_{j=1}^{M} [V_j V_j^T]^{I_{ij}}$$

$$\mu_i^* = [\Lambda_i^*]^(- 1) \left(\alpha \sum_{j=1}^{M} [V_j R_{ij}]^{I_{ij}} + \Lambda_U \mu_U \right)$$

Since the users are independent, the conditional probability of the latent user matrix can be factorized into the product of individual users. Bu this manner, the sampling process of the user vectors can be done in parallel.

$$p(U|R, V, \Theta_U) = \prod_{i=1}^{N} p(U_i|R, V, \Theta_U)$$

The user hyper-parameters conditional probability distribution on the user feature matrix U is the Normal-Wishart distribution:

$$p(\mu_U, \Lambda_U|U, \Theta_0) = \mathcal{N}(\mu_U|\mu_U^*, (\beta_0^*, \Lambda_U)^{-1}) \mathcal{W}(\Lambda_U|W_0^*, \nu_0^*)$$

where:

$$\mu_0^* = \frac{\beta_0 \mu_0 + N\bar{U}}{\beta_0 + N}, \beta_0^* = \beta_0 + N, \nu_0^* = \nu_0 + N$$

$$[W_0^*]^{-1} = W_0^{-1} + N\bar{S} + \frac{\beta_0 N}{\beta_0 + N} (\mu_0 - \bar{U})(\mu_0 - \bar{U})^T$$

$$\bar{U} = \frac{1}{N} \sum_{i=1}^{N} U_i, \bar{S} = \frac{1}{N} \sum_{i=1}^{N} U_i U_i^T$$

Since the conditional probability distributions over the movie latent feature vector and hyper-parameters will have the exact form, the Gibbs sampling algorithm to sample the latent user and movie feature is as follows: The next section summarizes the technical details of Bayesian Non-Negative Matrix Factorization model.

3.2.2 Bayesian Non-Negative Matrix Factorization

The details of the model can be found in the original work [60]. The same context and notations are used as in the last section. The non-negative matrix factorization can be depicted as:

$$R = U^T V + E$$

Algorithm 2 Gibbs sampling algorithm for BPMF

Initiate the latent factorized matrix component $\{U^1, V^1\}$. $U \in \mathbb{R}^{D\times N}$, $V \in \mathbb{R}^{D\times M}$.

for $t = 1$ to T **do**

1. Sample the hyper-parameters:

$$\mu_U^t, \Lambda_U^t \sim p(\mu_U, \Lambda_U | U^t, \Theta_0)$$
$$\mu_V^t, \Lambda_V^t \sim p(\mu_V, \Lambda_V | V^t, \Theta_0)$$

2. Sample $i = 1$ to N user vectors in parallel:

$$U_i^{t+1} \sim p(U_i | R, V^t, \mu_U^t, \Lambda_U^t)$$

3. Sample $j = 1$ to M movie vectors in parallel:

$$V_i^{t+1} \sim p(V_i | R, U^{t+1}, \mu_V^t, \Lambda_V^t)$$

end for

where $R \in \mathbb{R}^{N\times M}$ is the rating matrix, $U \in {\mathbb{R}_+}^{D\times N}$ and $V \in {\mathbb{R}_+}^{D\times M}$ are the factorized matrix of users and movies latent features, and $E \in \mathbb{R}^{N\times M}$ is the residual matrix. There is a non-negative constraint over the elements of the factorized matrices. Assume that the residuals E_{ij} is i.i.d and normal distributed with zero mean and variance σ^2. Then, the likelihood in the Bayesian model is:

$$p(R|\Theta) = \prod_{i,j} \mathcal{N}(R_{i,j}; (U_i^T V_j, \sigma^2)$$

where:

- $\Theta = \{U, V, \sigma^2\}$ is all the parameters in the model
- $\mathcal{N}(x; \mu, \sigma^2)$ is the pdf of a Normal distribution with mean μ and variance σ^2
- U and V are independently exponentially distributed with scales $\alpha_{d,i}$ and $\beta_{d,j}$

$$p(U) = \prod_{d,i} \mathcal{E}(U_{d,i}; \alpha_{d,i})$$
$$p(V) = \prod_{d,j} \mathcal{E}(V_{d,j}; \beta_{d,j})$$

where $\mathcal{E}(x;\lambda)$ is the pdf of a exponential distribution with rate (inverse scale) λ. The prior of σ^2 is chosen as an inverse gamma probability distribution with shape k and scale θ:

$$p(\sigma^2) = \mathcal{G}^{-1}(\sigma^2; k, \theta) = \frac{\theta^k}{\Gamma(k)}(\sigma^2)^{-k-1}\exp\left(-\frac{\theta}{\sigma^2}\right)$$

The graphical model of BNMF is: As in the last section, the conditional

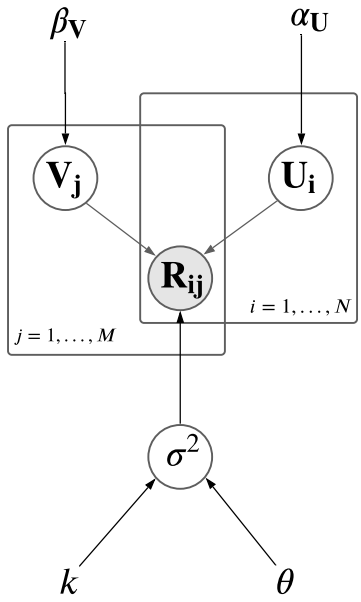


Figure 3.2: The graphical model of BNMF model

distributions are needed to sample by the Gibbs sampling algorithm. The conditional probability of the user $U_{d,i}$ is:

$$p(U_{d,i}|R, U_{\setminus(d,i)}, V, \sigma^2) = \mathcal{N}(U_{d,i}; \mu_{U_{d,i}}, \sigma^2_{U_{d,i}})\mathcal{E}(U_{d,i}; \alpha_{d,i})$$

where:

$$\mu_{U_{d,i}} = \frac{\sum_j (R_{i,j} - \sum_{d' \neq d} U_{d',i}V_{d',j})V_{d,j}}{\sum_j V^2_{d,j}}$$

$$\sigma^2_{U_{d,u}} = \frac{\sigma^2}{\sum_j V^2_{n,j}}$$

Due to symmetry, the conditional probability distribution of the movie $V_{d,j}$ has the same form. The conditional probability density of σ^2 is:

$$p(\sigma^2|R, U, V) = \mathcal{G}^{-1}(\sigma^2; k_{\sigma^2}, \theta_{\sigma^2})$$

where:

$$k_{\sigma^2} = \frac{NM}{2} + 1 + k$$
$$\theta_{\sigma^2} = \frac{1}{2}\sum_{i,j}(X - U^T V)^2_{i,j} + \theta$$

Since the users (and movies) are independent from each other. A parallel sampling algorithm can be derived. From those conditional probability distribution above, the Gibbs sampling algorithm for BNMF is presented in the Algorithm 3.

Algorithm 3 Gibbs sampling algorithm for BNMF

1. Initiate the latent factorized matrix component $\{U^1, V^1\}$. $U \in \mathbb{R}^{D\times N}$, $V \in \mathbb{R}^{D\times M}$.
2. $\chi = \frac{1}{2}\sum_{i,j} R^2_{ij}$

for $t = 1$ to T **do**
  $C = VV^T$, $D = RV^T$
  for $d = 1$ to D **do**
    $\mu_{U_d} = \frac{D_{d,:} - C_{d,\setminus d} U_{\setminus d,:} - \alpha_{d,:}\sigma^2}{C_{d,d}}$
    $U_{d,:} \sim \mathcal{N}(\mu_{U_d}, \frac{\sigma^2}{C_{d,d}})\mathcal{E}(\alpha_{d,:})$
  end for
  $\xi = \frac{1}{2}\sum_{d,i} U_{d,i}(AC - 2D)_{d,i}$
  $\sigma^2 \sim \mathcal{G}^{-1}(\frac{NM}{2} + k + 1, \chi + \theta + \xi)$
  $E = UU^T$, $F = RU^T$
  for $d = 1$ to D **do**
    $\mu_{V_d} = \frac{F_{d,:} - E_{d,\setminus d} V_{\setminus d,L} - \beta_{d,:}\sigma^2}{E_{n,n}}$
    $V_{d,:} \sim \mathcal{N}(\mu_{V_d}, \frac{\sigma^2}{E_{d,d}})\mathcal{E}(\beta_{d,:})$
  end for
  $U^t \leftarrow U$, $V^t \leftarrow V$
end for

Next, the following section explores how the Bayesian Matrix Factorization can be utilized to perform BO in the case of missing data.

3.3 Bayesian Optimization with Missing Inputs

In this section, the work of Luong et al. [42] is presented. The authors take advantage of the Bayesian Matrix Factorization model to perform BO with

missing data.

As laid out in the Algorithm 1, the black-box objective function is represented by a surrogate model (often a GP) through the data. However, problems arise when there are missing data. The authors choose the BPMF to model the missing data by finding the factorized components U and V such as $U^T V \approx X$. Then, the authors do not simply aggregate the information found by sampling U and V. Such as computing the average over the samples to obtain the imputation of the missing data. Instead, the samples of the factorized matrices U and V are kept to reconstruct the missing values.

Later, these imputations are combined with the original data to produce several imputed versions of the data. Then, these imputed versions are modeled by Gaussian Processes and later are utilized by an acquisition function, Upper Confidence Bound with Missing Inputs (UCB-MI), that consider those imputed versions to suggest the next point to evaluate. The evaluated point (which can be missing after evaluation) and the result are added to the original data, and the BO loop continues. The process is illustrated in the Algorithm 4.

The UCB-MI acquisition function is discussed in the next section.

3.3.1 Upper Confidence Bound with Missing Inputs

In other to incorporate information from several GPs to suggest the next point, the authors of [42] develop Upper Confidence Bound with Missing Inputs (UCB-MI), which aggregates information from individual acquisition functions to take into account the uncertainty of the imputed values. Suppose there a a set of GPs, $\{\mathcal{GP}^t\}_{t=1}^T$, as described in the Algorithm 4. The individual GP-UCB acquisition functions, α^t_{GP-UCB}, can be computed from these GPs (section 3.1.3.1). Then, those GP-UCB acquisition functions can be combined to form the UCB-MI as follows:

$$\alpha_{UCB-MI} = \mu_\alpha(\boldsymbol{\alpha}_{GP-UCB}(x)) + \beta_\alpha \sigma_\alpha(\boldsymbol{\alpha}_{GP-UCB}(x))$$

where:

$$\mu_\alpha(\boldsymbol{\alpha}_{GP-UCB}(x)) = \frac{1}{T}\sum_{t=1}^{T} \alpha^t_{GP-UCB}(x)$$

$$\sigma_\alpha(\boldsymbol{\alpha}_{GP-UCB}(x)) = \sqrt{\frac{\sum_{t=1}^{T}\left(\alpha^t_{GP-UCB}(x) - \frac{1}{T}\sum_{t=1}^{T}\alpha^t_{GP-UCB}(x)\right)^2}{T-1}}$$

We can see that:

Algorithm 4 Bayesian Optimization for Missing Inputs

Input data $X = \{x_n, f(x_n)\}_{n=1}^{N} \in \mathbb{R}^{N \times M}$
Let the set $\{i_k, j_k\}_{k=1}^{K}$ is the set of missing data indices in X.
Budget to run the experiment B.
for $b = 1$ to B **do**
1. Sampling T factorized matrices set $\{U^t, V^t\}_{t=1}^{T}$ of X by BPMF. $U \in \mathbb{R}^{D \times N}$, $V \in \mathbb{R}^{D \times M}$
for $t = 1$ to T **do**
$X^t \leftarrow X$
for $k = 1$ to K **do**
Reconstruct the missing data $x^t_{\{ij\}_k} = (U^t_{i_k})^T (V^t_{j_k})$
$X^t_{\{ij\}_k} \leftarrow x^t_{\{ij\}_k}$
end for
end for
2. The set $\{X^t\}_{t=1}^{T}$ contain the imputed version of X.
3. Model $\{X^t\}_{t=1}^{T}$ by the set of Gaussian Processes $\{\mathcal{GP}^t\}_{t=1}^{T}$.
4. Using the acquisition function UCB-MI with the set of Gaussian Processes $\{\mathcal{GP}^t\}_{t=1}^{T}$ to suggest the next point to evaluate x_{next}.
if *missing event* **then**
$x_{next} \leftarrow x'_{next}$.
end if
5. Compute $f(x_{next})$ then add $\{x_{next}, f(x_{next})\}$ to the original data X.
end for

- $\mu_\alpha(\boldsymbol{\alpha}_{GP-UCB}(x))$ represents the mean of the set of individual GP-UCB acqusition functions or the agreement between them.
- $\sigma_\alpha(\boldsymbol{\alpha}_{GP-UCB}(x))$ represents the standard deviation of them or the disagreement between them.
- And β_α is the trade-off between the agreement and disagreement.

. Finally, the next point can be obtained by:

$$x_{t+1} = \arg\max_{x \in \mathcal{D}} \alpha_{UCB-MI}(x)$$

The following section introduces the final tool in the toolbox to tackle the problem of limited budget in each experiment by using methods in the Multi-armed Bandit Framework.

3.4 Multi-armed Bandit Framework

The Multi-armed Bandit Framework is usually applied to reason about interactions between an agent and a stochastic environment where:

- The goal is to gain the maximum return/reward from the environment.
- There is a limited budget to interact with the environment.
- The agent needs to choose how to interact with the environment.
- Each interaction with the environment gives the agent feedback to decide between doing what is best (exploitation) or changing to another way of interaction (exploration). In the literature, the individual interaction/action is called "arm". When selecting that interaction/action, it is called "pulling an arm". These terminologies are used from now on.

The framework is helpful since only a limited number of experiment parameter settings can be chosen to maximize the experiment's results. The details will be presented in Chapter 4. The multi-armed bandit problem was first introduced by Robbins [56] and has been extensively researched since then. Some useful overview resource can be found in these works [34] [61]. In this work, only the two simplest algorithms of the multi-armed bandit framework are employed to limit the scope. They are ϵ-greedy and Upper Confidence Bound (UCB1). The next section describes those in detail.

3.4.1 ϵ-greedy

This is a very simple algorithm. While the budget is still available, at each iteration, the agent choose the arm that has the highest expected rewards so far with the probability $1-\epsilon$. And choose a random arm with the probability ϵ. In other words, if there are N arms, the probability of selecting arm i at iteration t is:

$$p_i(t) = \begin{cases} 1 - \epsilon + \epsilon/n & \text{if arm } i\text{'s expect reward is the highest} \\ \epsilon/n & \text{otherwise} \end{cases}$$

3.4.2 Upper Confidence Bound (UCB1)

The Upper Confidence Bound algorithm family has been proposed by Auer et al. [1]. The simplest version of the family, UCB1, is an improved version of ϵ-greedy. In UCB1, the number of times the arm has been pulled is considered.

The expected reward of an arm after t times the arm have been pulled is the exploitation factor. While the number of time that arm have been pulled with the total iteration so far contribute to the exploration factor. Specifically, suppose there are k arms, at round n the arm i is greedily picked by the formula:

$$i_n = \arg\max_{i=1\ldots k} \left(\hat{\mu}_i + \sqrt{\frac{2 \ln n}{t_i}} \right)$$

where $\hat{\mu}_i$ is the expected reward of arm i at round n. Moreover, t_i is the number of times the arm i has been pulled.

The usage of the ϵ-greedy, UCB1, and their contribution to our methods will be discussed in-depth in the subsequent chapters.

Chapter 4

Methods

This section first states this thesis problem formally, then presents the methods used to solve that problem.

4.1 Problem Definition

In summary, the problem of Bayesian Optimization with partially overlapping covariate data sources with constraint budgets is being tackled. Formally, suppose there is the set of T data set $\{X^t\}_{t=1}^T$ where each X^t has the set of covariates $C^t = \{c_1^t, c_2^t, ...\}$ and the set of response vector $\{y^t\}_{t=1}^T$. The data sources' covariates are partially overlapping which is $\forall i, j$, $C^i \neq C^j$, and $\exists i, j$ such that $C^i \cap C^j \neq \emptyset$. In theory, the method can work where all C^t are disjoint. However, the experiments conducted do not include such cases. Therefore, the overlapping covariate constraint is included.

Another condition is that each X^t has to represent the setting of the same process or mechanism f, and the response $y^t = f(x^t)$ must have the same meaning across X^t. For example, each X^t can be different methods to make beer, and each method has somewhat similar settings but is not identical. Furthermore, the response y^t should be the same beer evaluation score.

Given the data source set $\{X^t\}_{t=1}^T$ and the response vector set $\{y^t\}_{t=1}^T$, the problem is to find the setting x that gives the maximum return, with a limited amount of experiment can b run, from the process, or a black-box function f:

$$x^* = \arg\max_{x \in \mathcal{D}} f(x)$$

where $\mathcal{D}$ is the domain of x

One important detail is that the covariates of x^* is not the union $\bigcup_{t=1}^T C^t$ of the set of the covariates, but only a small subset of it, i.e., $C^{x^*} \subset \bigcup_{t=1}^T C^t$,

where C^{x^*} is the set of covariates of x^*. The covariates of those data sources do not coincidentally partially overlap. It is the consequence of choosing some parameters to try in different experiment settings because of the budget or technology constraints. Also, separated experiments will have different sets of parameters.

For example, breweries will have different formulae due to the availability of the ingredients, traditions, or technologies. Moreover, if there is a need to open a new brewery and try to utilize the data of others around the globe, it is implausible that one can obtain the union of all ingredients, techniques, and technologies of all other breweries combined. The same argument can be applied in the case of industrial or agricultural processes. Therefore, there are two budgeting constraints in this work. The usual constraint of the number of experiments can be run, and the constraint of number parameters can be changed in each experiment.

The following section presents the primary method to solve the above problems formally.

4.2 Bayesian Optimization for Partially Overlapping Covariate Data Sources with Constraint Budget

In this section, the method to tackle the problem of Bayesian Optimization for partial overlapping covariate data sources with a constraint budget is presented. Based on the problem definition in Section 4.1, the method needs to satisfy three requirements:

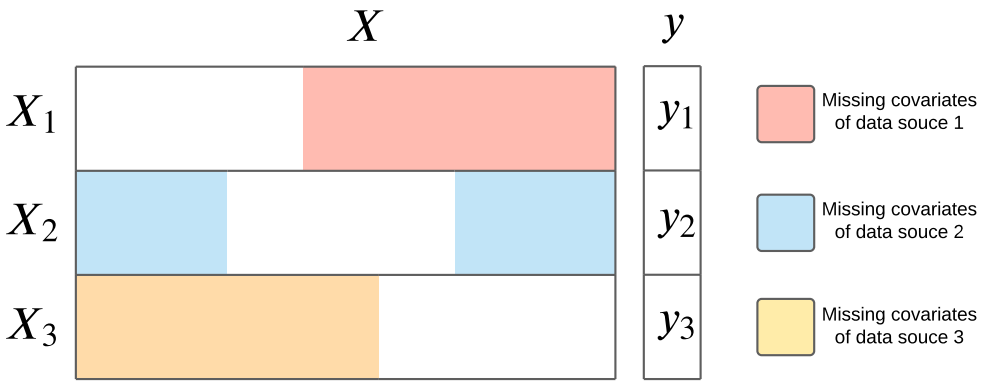
1. The method needs to utilize the information from the partially overlapping data sources $\{X^t\}_{t=1}^T$.

2. By utilizing the data, the method needs to find the experiment setting x^* maximizing the black-box function's return with a limited number of experiments.

3. In each experiment, only some experiment settings can be tuned/changed, and the others are left at default or simply unobserved/uncontrollable. The method needs to take into account this constraint.

In order to address the first two requirements, the data source set $\{X^t\}_{t=1}^T$ is stacked row-wise to form the master data set X where there are missing data in X. The missing data are the experiment settings that do not present

in one data source but present in others. The stacked data set X is illustrated by the Figure 4.1. The problem now becomes to suggest the next



How to utilize data sets with overlapping covariates/features in Bayesian Optimization?

Figure 4.1: Data sources X_1, X_2, X_3 are stacked to form the master data source with missing data X

experiment given a missing data set. This problem can be addressed by applying Bayesian Optimization for Missing Inputs that have been laid out in Section 3.3. After applying the BOMI method to the missing data X, the setting x_{next} can be obtained and can be evaluated to find the $y_{next} = f(x_{next})$. Then the BO loop continues.

However, the third requirement demand that only a subset of the experiment settings of x_{next} can be chosen to tune or change and let the others vary at random or unobserved. In order to balance the immediate and future reward in choosing the combination of settings of x_{next}, the multi-armed bandit model can be applied in this situation. Each combination of the experiment settings of x_{next} can be considered as an arm. Each time a combination of experiment settings is chosen, this can be seen as pulling an arm, and the return $f(x_{next})$ is the reward for pulling that arm.

In short, by applying the BOMI and multi-armed bandit methods, an algorithm to tackle the problems stated in Section 4.1 is proposed. The overview of the method is shown in Figure 4.2, and the details are presented in the Algorithm 5. The following section goes through the implementation of this method and how we set up the experiments.

Algorithm 5 Bayesian Optimization for Partial Overlapping Covariate Data Sources with Constraint Budget

a. Input is the set of data source $\{X^t\}_{t=1}^T$, which covariates are partially overlap, with the set of response vector $\{y^t\}_{t=1}^T$, that describe an underlying black-box process f.
b. Stack X^t column-wise to form the missing data set X.
c. Let the set $\{i_k, j_k\}_{k=1}^K$ is the set of missing data indices in X. $X \in \mathbb{R}^{N \times M}$.
d. Maximum number of experiments can run is B.
e. Maximum number of parameters can change in each experiment P.
for $b = 1$ to B **do**
1. Sampling S factorized matrices set $\{U^s, V^s\}_{s=1}^S$ of X by a Bayesian Matrix Factorization method. $U \in \mathbb{R}^{D \times N}$, $V \in \mathbb{R}^{D \times M}$, where $X = U^T V$.
for $s = 1$ to S **do**
$X^s \leftarrow X$
for $k = 1$ to K **do**
Reconstruct the missing data $x^s_{\{ij\}_k} = (U^s_{i_k})^T (V^s_{j_k})$
$X^s_{\{ij\}_k} \leftarrow x^s_{\{ij\}_k}$
end for
end for
2. The set $\{X^s\}_{s=1}^S$ contain the imputed version of X.
3. Model $\{X^s\}_{s=1}^S$ by the set of Gaussian Processes $\{\mathcal{GP}^s\}_{s=1}^S$.
4. Using the acquisition function UCB-MI with the set of Gaussian Processes $\{\mathcal{GP}^s\}_{s=1}^S$ to suggest the next point to evaluate x_{next}.
5. Select the set of P parameter settings/covariates $\mathbf{c} = \{c_p\}_{p=1}^P$ to change in x_{next} based on a Multi-armed Bandit Algorithm (e.g. UCB1). Other parameters are set randomly (unobserved) by the process f.
6. Compute $f(x^{\mathbf{c}}_{next})$ with partially observed $x^{\mathbf{c}}_{next}$ then add $\{x^{\mathbf{c}}_{next}, f(x^{\mathbf{c}}_{next})\}$ to the original data X.
7. Update the reward (in multi-armed bandit algorithm) of the set of covariates $\mathbf{c}$: $\mathbf{r_c} = \mathbb{E}[f(x^{\mathbf{c}}_{next})]$ where $\mathbb{E}[f(x^{\mathbf{c}}_{next})]$ is the expectation reward of $x^{\mathbf{c}}_{next}$ up to iteration b.
end for

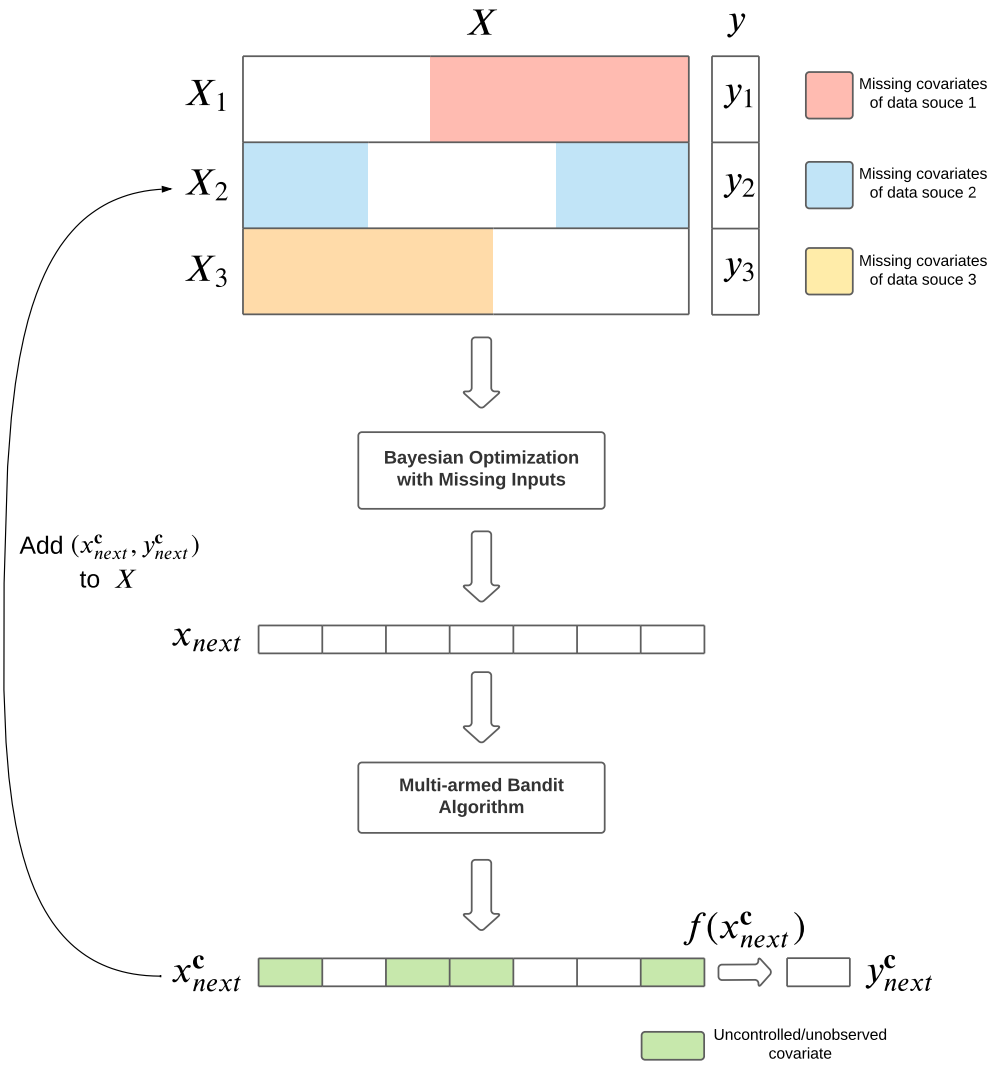


Figure 4.2: Overview of Bayesian Optimization for Partial Overlapping Covariate Data Sources with Constraint Budget

Chapter 5

Implementation

This chapter goes through the setup of the experiments to test the method introduced in Chapter 4. The software and the implementation used to realize that method are also described.

5.1 Experimentation

There are two experiments to simulate the black-box function or an industrial/agriculture process f: the training process of a Random Forest Classifier and manufacturing cellulose-based carbon fibers.

5.1.1 Random Forest Classifier Experiment

In the first experiment, the black-box function f is simulated by the process of training a Random Forest Classifier on the Covertype Data Set in the UCI Machine Learning Repository [15].

The train and test split of the data set is fixed to consistently assess the performance of different classifier configurations in the proposed Bayesian Optimization method. The accuracy of the classifier is chosen to be the return $y = f(x)$, and x is the different hyper-parameters of the Random Forest Classifier in scikit-learn.

In each experiment iteration, hyper-parameters that need to be changed are chosen according to the Multi-armed Bandit Algorithm in the proposed method. The others are unobserved and will be randomly assigned by the black-box function. It is reasonable to assume that those unobserved hyper-parameters will vary around default values under normal conditions. In order to simulate desired behavior, the unobserved hyper-parameters are set to their default values and are varied in a set of predefined ranges.

The hyper-parameters, their value ranges, default values, varied ranges, and their effects on the complexity of the classifier, when increased, are summarized in Table 5.1. The differences in value ranges, default values, the varied ranges from the defaults, and the effect on the model complexity of each parameter will add to the difficulty of the optimization process, which helps in emulating a real-world process better.

Hyper-parameter	**Value Range**	**Default Value**	**Varied Range (from default)**	**Model Complexity Effect**
n estimators	(10, 2000)	100	(1, 20)	Increase
max depth	(5, 10000)	None	(100, 1000)	Increase
min samples split	(2, 100)	2	(1, 10)	Decrease
min samples leaf	(1, 100)	1	(1, 10)	Decrease
min weight fraction leaf	(0, 0.01)	0.0	(0.001, 0.005)	Decrease
max leaf nodes	(100, 1000)	None	(20, 500)	Increase
min impurity decrease	(0, 0.001)	0.0	(0.0001, 0.0005)	Decrease

Table 5.1: Hyper-parameters' settings of the Random Forest Classifier experiment

The initial data for this experiment is shown in Table 5.2. The initial data emulate stacked data set from three different data sources that overlap at one covariate. Furthermore, each experiment from each data source has only three parameter settings available. The observed parameter settings are varied around the default value around a predefined range shown in Table 5.1. The classification accuracy of each experiment is also presented.

n esitmators	**max depth**	**min samples split**	**min samples leaf**	**mean weight fraction leaf**	**max leaf nodes**	**min impurity decrease**	**Classification Accuracy**
10	10	3					0.7016
25	30	10					0.7174
50	100	20					0.7302
		25	1	0.003			0.7132
		14	3	0.0015			0.7131
		4	7	0.002			0.7137
				0.004	50	0.00015	0.7098
				0.0031	25	0.00025	0.6936
				0.0022	75	0.00035	0.7076

Table 5.2: Initial Data of the Random Forest Experiment

In this experiment, the number of times new parameter settings can be tested is set at 200. The number of parameter settings that can be chosen to test (and leave others unobserved) ranges from $[3 \dots 7]$ where 3 is the

minimum number of settings that each experiment can have before stacking them together, and 7 is the maximum number of settings after stacking.

In short, this experiment aims to find the experiment settings that maximize the classification accuracy after 200 iterations using $[3 \dots 7]$ settings each time. The results are presented in Section 6.1. The next section describes the experiment using the carbon fiber manufacturing process data.

5.1.2 Carbon Fiber Manufacturing Experiment

The carbon fiber manufacturing data is collected from the paper of Zhang et al. [80] by the company Teraloop[1]. The data set has 32 data points. The covariates of the data set are High T2 Temperature (Celsius), Tension (MPa), Tensile Strength (GPa), Strain at break (%), and Young's Modulus (GPa). Young's Modulus is chosen to be the response $y = f(x)$, and others are parameter settings x.

Six random data points (except the data point with highest Young's modulus, explain later) are chosen at random with some covariates remove to simulate a stacked data set formed from different partial overlapping sources. The initial data is presented in Table 5.3.

High T2 Temperature (Celsius)	**Tension (MPa)**	**Tensile Strength (GPa)**	**Strain at break (%)**	**Young's Modulus (GPa)**
2100	36			50
2100	72			50
	108	1.22		66
	144	1.14		68
		1.15	1.78	63
		1.22	1.6	75

Table 5.3: Initial Data of the Carbon Fiber Manufacturing Experiment

Since the underlying process to evaluate the new experiment settings is not available, the 26 remaining data points are used to imitate the carbon fiber manufacturing process. Specifically, after modeling the training data with Gaussian Processes to form a surrogate model, the acquisition function runs through the remaining data points and chooses the one with the highest acquisition score to be the next experiment to evaluate. The budgeting of the experiment settings, i.e., selecting only some of the experiment settings

[1] https://www.teraloop.org/

to evaluate, is kept unchanged. The budgets to choose parameter settings to change at each iteration are set in the range $[2 \dots 4]$.

The budgeting step now has a different meaning. In the Random Forest experiment, the experiment setting budgeting restricts the information from the suggested experiment go into the underlying black-box process. Hence, it undermines the ability of Bayesian Optimization to maximize the return. Consequently, the Multi-armed Bandit Algorithm is needed to alleviate that problem. Since the underlying carbon manufacturing process is not available in this experiment, the return is retrieved from the past data with all the covariates observed.

Nevertheless, experiment setting budgeting still has the meaning of limited information and causes finding the maximum return harder. Now, the meaning of budgeting the suggested experiment becomes limiting the information goes into the next iteration. Therefore, the utility of a Multi-armed Bandit Algorithm can still be applied.

Because of the lacking of the underlying black-box process, the goal of this experiment is different from the former. Instead of finding the setting that yields the maximum return, the target is now finding the setting with the least amount of experiment run. Because there is a finite set of data points, a data point with the maximum return can always be found. That also explains why the data point with a maximum return is not chosen in the initial data set. The results of this experiment can be found in Section 6.2.

The following section describes the parameters of the components of the proposed method.

5.2 Method's Components' Parameters

Since many configurations can be made in each of these components, the most parameters that impact the problem are controlled and described here. Others are kept based on the suggestion in the original papers. The details of the components and their origin can be found in Chapter 3. The components and their configurations are:

The Bayesian Matrix Factorization: There are two tested methods, BPMF and BNMF. Apart from the default parameters of each method, the controlled parameters are the number of latent variable D, the number of Gibbs sampling algorithm iterations G, and the number of samples S of factorized matrices U and V. We choose $D = 15$, $G = 200$, and $S = 5$. The samples of factorized matrices U and V are obtained at the end of the Gibbs sampling process.

The UCB and UCB-MI Acquisition Function: We set the $a, b = 1$ in UCB, and $\beta_\alpha = 0.2$ in UCB-MI.

Multi-armed Bandit Algorithms: The experiments are evaluated by two algorithms, ϵ-greedy and UCB1. In ϵ-greedy, the two settings $\epsilon = 0.1$ and $\epsilon = 0.3$ are tried. There is no parameter to config in the UCB1.

5.3 Software

According to Figure 4.2, there are two main components in our methods. The components and the software for those are listed below:

- The Bayesian Optimization for Missing Inputs (BOMI) to deal with the stacked data set X and limited experiment runs. The implementation of the original paper [42] is used. It can be accessed in their GitHub repository [40].
- The Multi-armed Bandit Algorithm to select the potential high-reward experiment settings. We implemented two Multi-armed Bandit Algorithms ourselves, which are ϵ-greedy and UCB1.

In BOMI, as described in Section 3.3, there are also two main components:

- The first one is the Bayesian Matrix Factorization to handle the missing data. In the original BOMI paper, the author used the BPMF implementation of Pacchiardi, which can be accessed in their GitHub repository [51]. The method BNMF is employed to compare with BPMF. The BNMF implementation of Brouwer is used. It has been published in the paper [6] and also has a GitHub repository [5].
- The second one is the UCB-MI acquisition function to incorporate the uncertainty of imputed data to suggest the next experiment to perform. The implementation of Luong [40] mentioned above is employed.

Apart from these main components above, the Python 3.7.9 and other common data science packages/tools such as scikit-learn [54], NumPy [23], pandas [52], and BoTorch [2] are used in the implementation of the proposed method.

Chapter 6

Evaluation

This chapter presents the results of this thesis.

6.1 Random Forest Classifier Experiment

Figure 6.1, Figure 6.2 shows the results of applying two Bayesian Matrix Factorization methods (BNMF, BPMF) and Multi-armed Bandit Algorithms (ϵ-greedy, UCB1) to the Random Forest Classifier Experiment in the case of selecting three and four experiment settings. In the case of selecting only three experiment settings per iteration (three budget), BPMF combined with UCB1 achieves the best results. In the same case, in general, BPMF is a better model to incorporate the uncertainty of the missing data since it helps find better experiment settings sooner (required less iteration) than BNMF. In terms of the bandit algorithms, 0.1-greedy does better when combined with BPMF and BNMF and achieves the best result. Nevertheless, UCB1 consistently gives better results early on (less than 100 iterations) but can plateau later.

It is even more apparent in the case of selecting four settings (per experiment). Regardless of the bandit algorithms, the BPMF combinations significantly better results than BNMF. The UCB1 consistently provides significantly better results very early on, in this case, less than 50 iterations. Moreover, the algorithm is on par with 0.1-greedy which gives the highest return. The same can be said in the case of selecting five experiment settings in Figure 6.3, modeling the missing data by BPMF still provides an outstanding performance compared to BNMF.

In terms of the bandit algorithms, the advantage of UCB1 in the low iteration cases is no longer there, and other algorithms have caught up. A reasonable explanation that is because of the increase of available information

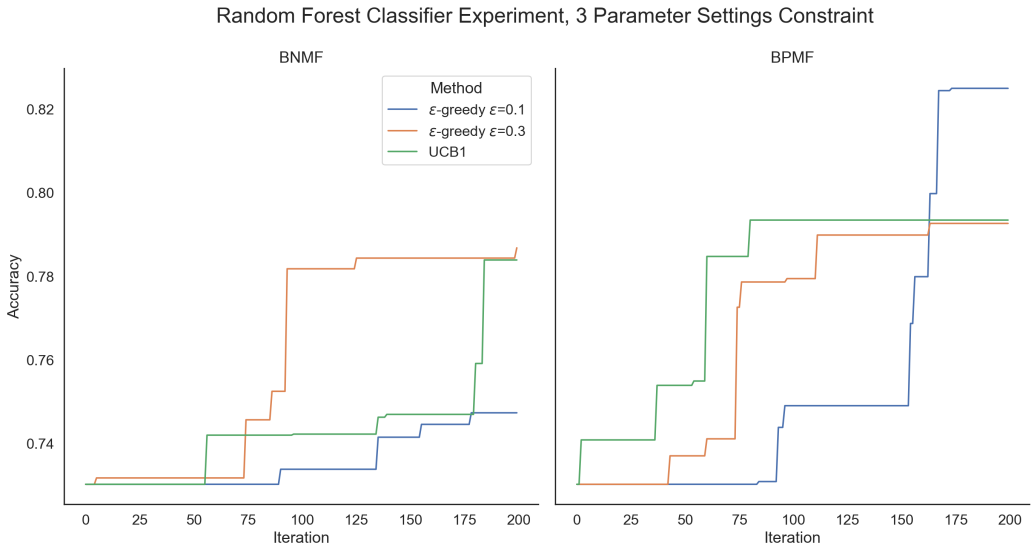


Figure 6.1: Random Forest Classifier Experiment, three parameter settings are selected each iteration.

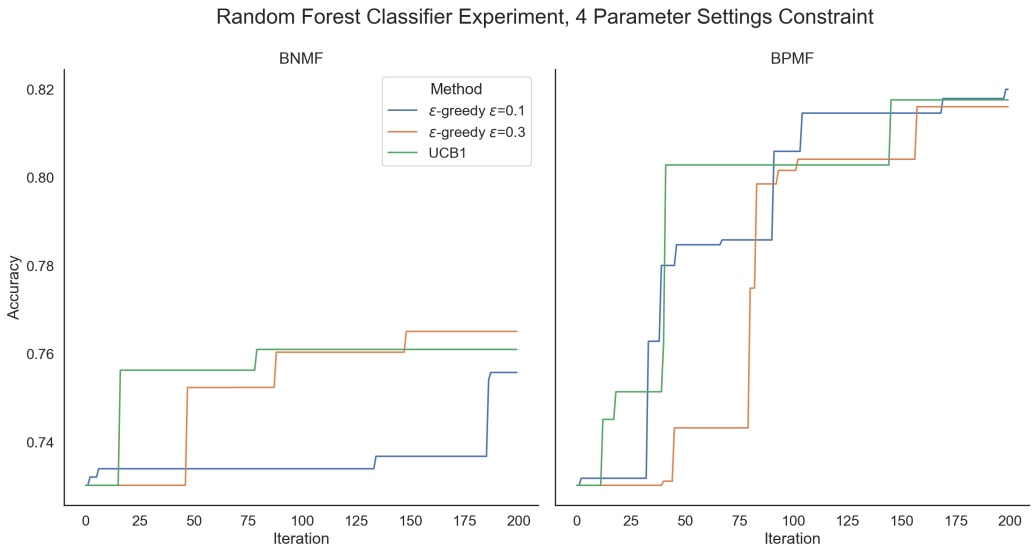


Figure 6.2: Random Forest Classifier Experiment, four parameter settings are selected each iteration.

in each iteration.

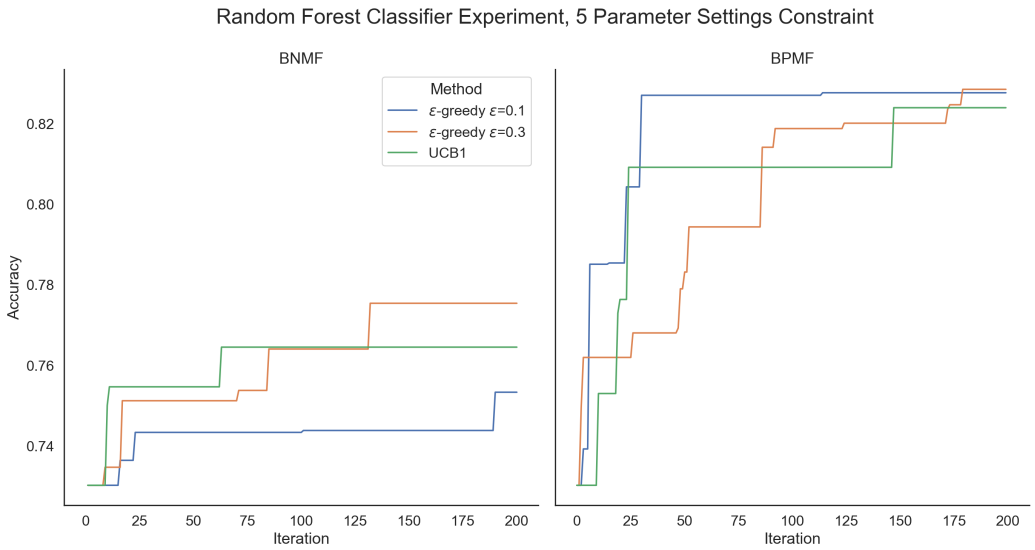


Figure 6.3: Random Forest Classifier Experiment, five parameter settings are selected each iteration.

Likewise, the increase of available information in each iteration is the reason the gap between the performance of BNMF-based methods and BPMF-based methods have been narrow down. Although there is still room for improvement for the BNMF-based methods, the evidence can be seen in Figure 6.4.

Finally, the results when not applying any budget constraint on the experiment settings are presented in Figure 6.5. The BPMF clearly outperforms the BNMF. The non negative condition on factorized element values of the BNMF may restrict the modeling capability which leads to this result.

6.1.1 Information Usage Effectiveness

This section introduces the concept of information usage effectiveness to compare these methods more generally. The effectiveness of the information usage can express how each method uses the information provided to find the maximum returns. Due to the parameter setting constraint at each iteration, the amount of information each method can use is different. Therefore, they must be normalized into an information unit to compare them more objectively. The information unit is defined as a single setting in a single experiment. If, at each iteration, one is allowed to choose three experiment

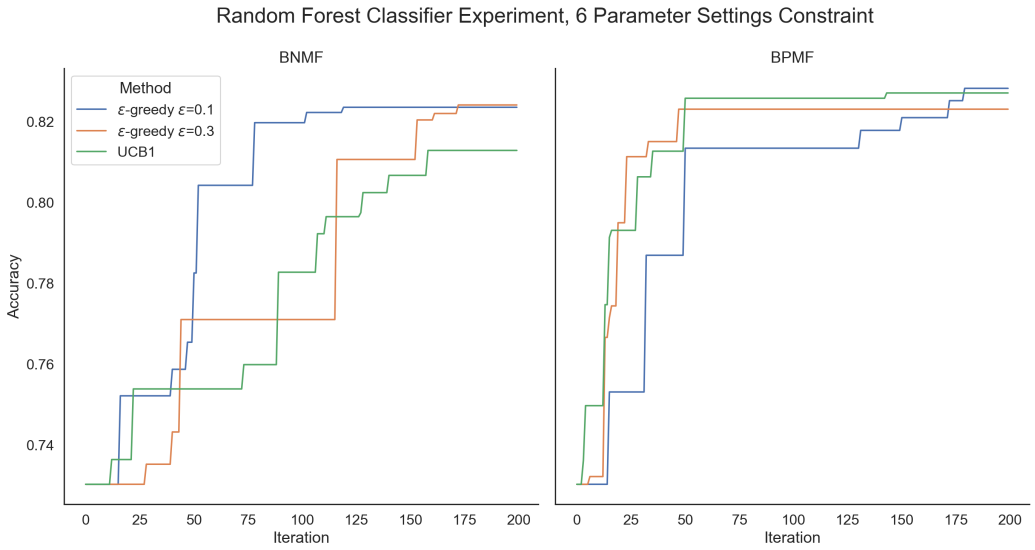


Figure 6.4: Random Forest Classifier Experiment, 6 parameter settings are selected each iteration.

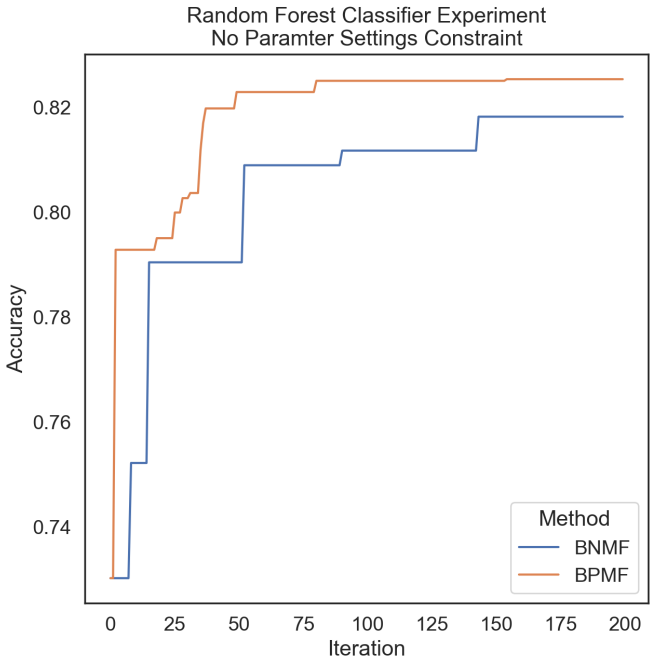


Figure 6.5: Random Forest Classifier Experiment, no constraint on selecting parameter settings.

settings to change and add to the data in the next iteration, the number of the information unit is increased by three.

In order to systematically assess the information usage effectiveness, the results of ϵ-greedy algorithms are not examined since they rely heavily on randomness. The UCB1 algorithm combined with the two Bayesian Matrix Factorization methods is chosen. The results can be seen in Figure 6.6.

The figure shows that the BPMF-based methods utilize the provided information more effectively than the BNMF-based methods. The BNMF-based methods can only use the information as effectively as BPNM-based methods with no parameter settings constraint in each iteration (no budgeting case). Again, the non-negative constraint of BNMF may lead to this result. However, this is just speculation and needs further analysis.

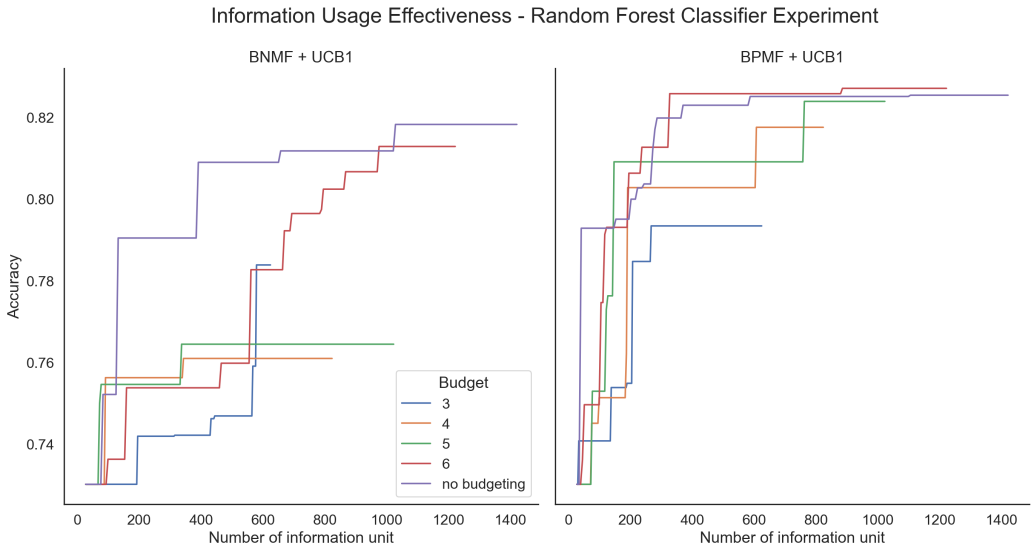


Figure 6.6: Information Usage Effectiveness, Random Forest Classifier Experiment.

The following section examines the carbon fiber manufacturing experiment.

6.2 Carbon Fiber Manufacturing Experiment

Similar to the previous section, we first inspect how the parameter setting constraint affects each method's performance. As explained in Section 5.1.2, due to the lack of a real-world carbon fiber manufacturing process, the historical data have been used to imitate the underlying mechanism. Therefore,

the performance criterion is not to maximize the return but to choose the record with the maximum return in the data set first through the missing data modeling and acquisition function.

In the case of selecting the two-parameter setting constraint in Figure 6.7, the 0.3-greedy achieves the best result when combined with BNMF. The performance profiles of 0.1-greedy and UCB1 are entirely the same under the BNMF modeling (A few jitters are added to distinguish between the two). It can be due to the limited amount of information in the data combined with the non-negative constraint of the BNMF. BPMF-based methods perform relatively well in the low iteration cases.

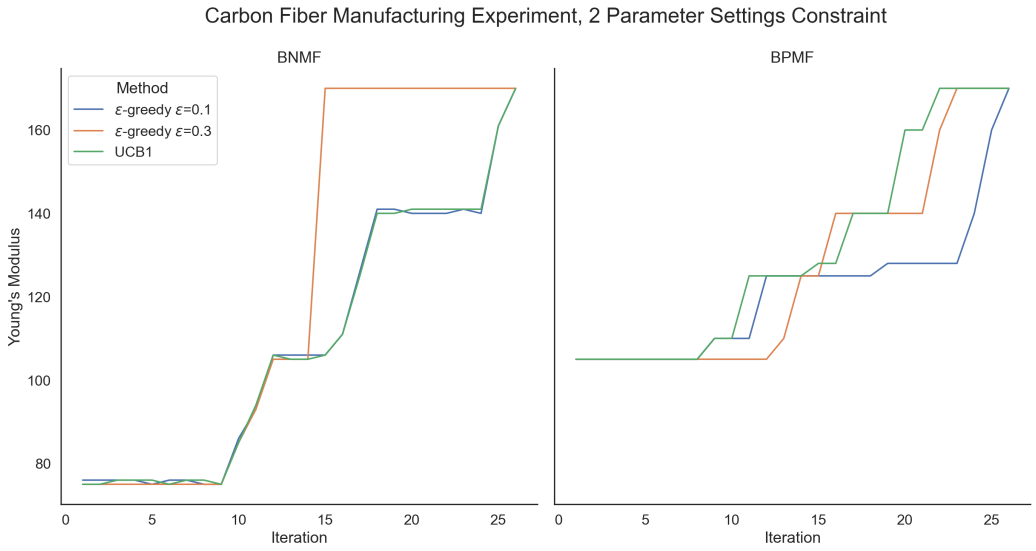


Figure 6.7: Carbon Fiber Manufacturing Experiment, 2 budget

BNMF-based methods show surprisingly good results in the case of selecting the three-parameter setting per iteration in Figure 6.8. These methods can select the setting that yields the highest return in the early iterations (less than 10) regardless of the bandit algorithm used. The reason may be that BNMF can somehow exploit the structure of the missing data set. Further investigation is needed to find the reason. BPMF-based methods still perform well, and UCB1 finds the highest yield setting the earliest.

The Figure 6.9 shows the result of removing the effect of parameter setting budgeting in each iteration. BPMF shows a notable better result than BNMF that is similar in the case of Random Forest Classifier Experiment. The yield line of BPMF is above the yield line of BNMF completely.

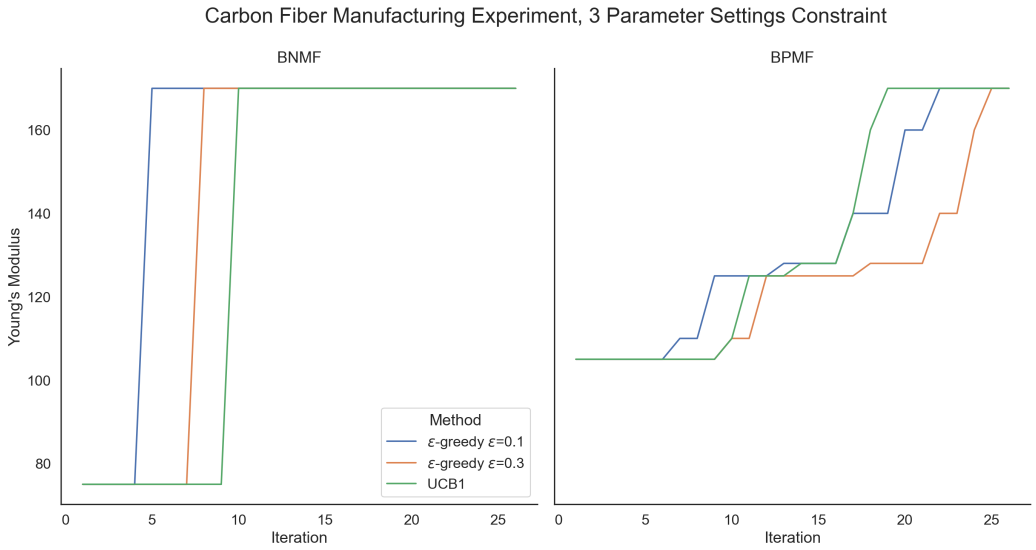


Figure 6.8: Carbon Fiber Manufacturing Experiment, three budget

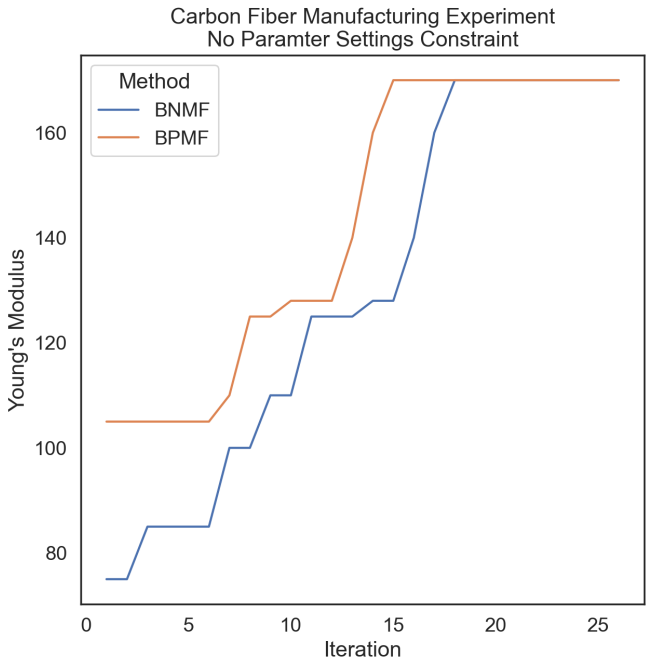


Figure 6.9: Random Forest Classifier Experiment, No Budgeting

6.2.1 Information Usage Effectiveness

Similar to the Section 6.1.1, the information usage effectiveness of this experiment is shown in the Figure 6.10. The results are very different from the Random Forest Classifier Experiment. The higher returns are not associated with the higher number of the information unit. For example, in the case of BPMF (and UCB1), the case of budgeting two parameter settings in each iteration gives a better result than the cases with more information. A similar situation can be seen in the case of the BNMF-based method. The three parameter settings budgeting gives a significantly better result than others. More experiments and analyses are needed to explain this phenomenon.

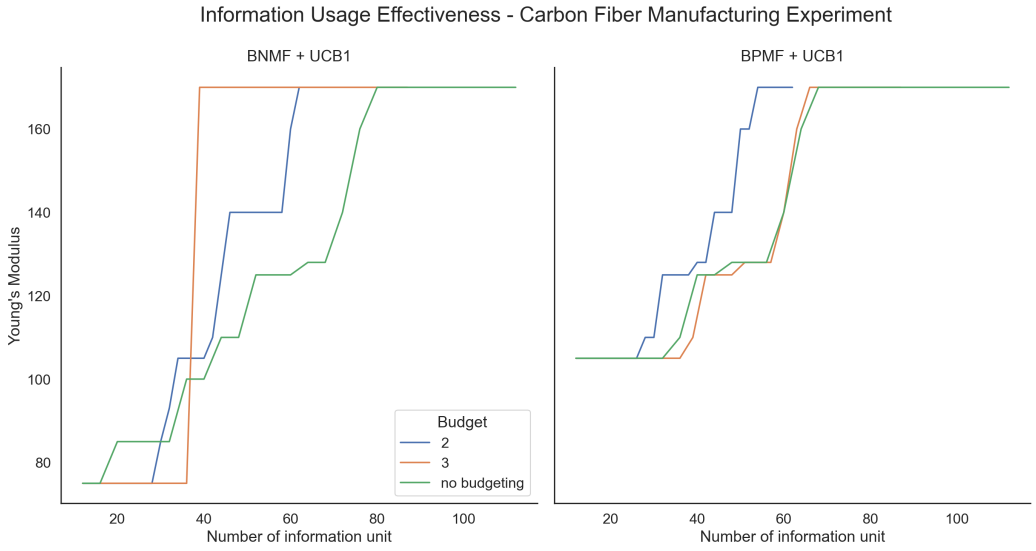


Figure 6.10: Information Usage Effectiveness, Carbon Fiber Manufacturing Experiment

The next chapter discusses the drawbacks and other aspects of the proposed methods. Future study directions to improve the current work are also proposed.

Chapter 7

Discussion

The problem definition and the proposed methods have been laid out in the previous chapters. The results have shown that the chosen approaches can address specific aspects of the problem. The effectiveness of the proposed methods has been demonstrated by applying them to two scenarios: one, the Random Forest Classifier, a simulated data scenario to replicate the behaviors of a black-box function, and two, a carbon fiber manufacturing process that uses data from a real-world to simulate the black-box function. Both have shown exciting and promising results.

Nevertheless, there are drawbacks related to the proposed approach, which are addressed in the following sections.

7.1 The Drawbacks

The major drawbacks of this work are:

- The number of iterations to find the parameter settings that can improve return is still high in the low information case. For example, in the Random Forest Classifier experiment, the number of iterations needed to get to a notable improvement in accuracy is around 75 to 100. Real-world industrial/agricultural processes often do not have the luxury to conduct that high number of experiments.
- The experiments in this study used a machine learning training process and past data to emulate a real-world production process that can hide the complexity of running production in reality.
- Some unexpected behavior of the methods is not fully explained.

Other minor drawbacks, e.g., different parameter settings, are not systematically tested in the experiments. Such concerns are mentioned in the future work directions in the next section.

7.2 Future Work Directions

To improve the presented method, the following lines of work could be beneficial:

- Internal structure of the data can be further exploited to pair with the appropriate missing data modeling methods. This idea has been mentioned briefly in Section 6.2.
- A real-world setting is needed to test the proposed method.
- Other models of the missing data or partial overlapping data can be studied. (e.g. the Bayesian Group Factor Analysis [75])
- Different Markov Chain Monte Carlo (MCMC) methods, apart from Gibbs sampling algorithm, such as Metropolis-Hastings [13] or Hamiltonian Monte Carlo [16][48] can also help to improve the factorized results. The diagnostics of the MCMC results can also be performed more deliberately.
- Other Multi-armed bandit algorithms such as Thompson Sampling [57], Pursuit Algorithm [70], or Reinforcement Comparison [68] can also be investigated.
- In the current method, the "arm", in the context of the Multi-armed bandit algorithm, is defined as the set of the experiment settings chosen to observe in each Bayesian Optimization iteration. This definition ignores the current values of the experiment settings. If two arms have the same parameter settings but at different values, they are still considered as the same "arm". Since the values of the settings significantly affect the result of the method, clever way to model the parameter settings with their values without generating too many numbers of "arm" can be very beneficial.

Chapter 8

Conclusions

This work has defined and proposed the method to solve the "Bayesian Optimization for Partially Overlapping Covariate Data Sources". The method has been tested in two experiments, a Random Forest Classifier experiment, and a Carbon Fiber Manufacturing experiment. The methods and the experiments have shown that the partially overlapping covariate data sources can be stacked row-wise and model the problem as a Bayesian Optimization with missing data. Furthermore, the methods have demonstrated the usage of Multi-armed bandit algorithms in solving the budget constraint of choosing the experiment settings. Both have shown improvements when the proposed methods are applied. Moreover, intriguing phenomena have been observed and briefly analyzed.

Although there are many major and minor drawbacks in the setup of the experiment and implementation, the method's general direction is promising and has opened many worth investigating new research questions. The method, with slight modification, can be translated into a process that can be very beneficial in real-world industrial and agricultural processes and experiments. That is also the goal of this work. Several future directions that can enhance this work further have been presented in Section 7.2.

Bibliography

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

[2] Balandat, M., Karrer, B., Jiang, D. R., Daulton, S., Letham, B., Wilson, A. G., and Bakshy, E. BoTorch: A Framework for Efficient Monte-Carlo Bayesian Optimization. In *Advances in Neural Information Processing Systems 33* (2020).

[3] Bennett, J., Lanning, S., et al. The netflix prize. In *Proceedings of KDD cup and workshop* (2007), vol. 2007, New York, NY, USA., p. 35.

[4] Breiman, L. Random forests. *Machine learning 45*, 1 (2001), 5–32.

[5] Brouwer, T. Fast bayesian nonnegative matrix factorisation and tri-factorisation. `https://github.com/ThomasBrouwer/BNMTF`, 2016. Online; accessed 14.01.2022.

[6] Brouwer, T., Frellsen, J., and Lio, P. Fast bayesian non-negative matrix factorisation and tri-factorisation. *arXiv preprint arXiv:1610.08127* (2016).

[7] Calvin, J., et al. One-dimensional global optimization for observations with noise. *Computers & Mathematics with Applications 50*, 1-2 (2005), 157–169.

[8] Calvin, J., and Žilinskas, A. On the convergence of the p-algorithm for one-dimensional global optimization of smooth functions. *Journal of Optimization Theory and Applications 102*, 3 (1999), 479–495.

[9] Calvin, J., and Žilinskas, A. One-dimensional p-algorithm with convergence rate o (n- 3+ δ) for smooth functions. *Journal of optimization theory and applications 106*, 2 (2000), 297–307.

[10] Calvin, J. M. Average performance of a class of adaptive algorithms for global optimization. *The Annals of Applied Probability* (1997), 711–730.

[11] Chang, J., Nikolaev, P., Carpena-Núñez, J., Rao, R., Decker, K., Islam, A. E., Kim, J., Pitt, M. A., Myung, J. I., and Maruyama, B. Efficient closed-loop maximization of carbon nanotube growth rate using bayesian optimization. *Scientific Reports 10*, 1 (2020), 1–9.

[12] Chaudhuri, A., Jasa, J., Martins, J. R., and Willcox, K. E. Multifidelity optimization under uncertainty for a tailless aircraft. In *2018 AIAA Non-Deterministic Approaches Conference* (2018), p. 1658.

[13] Chib, S., and Greenberg, E. Understanding the metropolis-hastings algorithm. *The american statistician 49*, 4 (1995), 327–335.

[14] Dempster, A. P., Laird, N. M., and Rubin, D. B. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society: Series B (Methodological) 39*, 1 (1977), 1–22.

[15] Dua, D., and Graff, C. UCI machine learning repository, 2017.

[16] Duane, S., Kennedy, A. D., Pendleton, B. J., and Roweth, D. Hybrid monte carlo. *Physics letters B 195*, 2 (1987), 216–222.

[17] Eggensperger, K., Feurer, M., Hutter, F., Bergstra, J., Snoek, J., Hoos, H., Leyton-Brown, K., et al. Towards an empirical foundation for assessing bayesian optimization of hyperparameters. In *NIPS workshop on Bayesian Optimization in Theory and Practice* (2013), vol. 10.

[18] Frazier, P., Powell, W., and Dayanik, S. The knowledge-gradient policy for correlated normal beliefs. *INFORMS journal on Computing 21*, 4 (2009), 599–613.

[19] Frazier, P. I. A tutorial on bayesian optimization. *arXiv preprint arXiv:1807.02811* (2018).

[20] Gardner, J., Guo, C., Weinberger, K., Garnett, R., and Grosse, R. Discovering and exploiting additive structure for bayesian optimization. In *Artificial Intelligence and Statistics* (2017), PMLR, pp. 1311–1319.

[21] Goldberg, K., Roeder, T., Gupta, D., and Perkins, C. Eigentaste: A constant time collaborative filtering algorithm. *information retrieval 4*, 2 (2001), 133–151.

[22] Gong, C., Peng, J., and Liu, Q. Quantile stein variational gradient descent for batch bayesian optimization. In *International Conference on Machine Learning* (2019), PMLR, pp. 2347–2356.

[23] Harris, C. R., Millman, K. J., van der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N. J., Kern, R., Picus, M., Hoyer, S., van Kerkwijk, M. H., Brett, M., Haldane, A., del Río, J. F., Wiebe, M., Peterson, P., Gérard-Marchant, P., Sheppard, K., Reddy, T., Weckesser, W., Abbasi, H., Gohlke, C., and Oliphant, T. E. Array programming with NumPy. *Nature 585*, 7825 (Sept. 2020), 357–362.

[24] Hayashi, S., Honda, J., and Kashima, H. Bayesian optimization with partially specified queries. *Machine Learning* (2022), 1–30.

[25] Hennig, P., and Schuler, C. J. Entropy search for information-efficient global optimization. *Journal of Machine Learning Research 13*, 6 (2012).

[26] Hernández-Lobato, J. M., Hoffman, M. W., and Ghahramani, Z. Predictive entropy search for efficient global optimization of black-box functions. *arXiv preprint arXiv:1406.2541* (2014).

[27] Huang, D., Allen, T. T., Notz, W. I., and Miller, R. A. Sequential kriging optimization using multiple-fidelity evaluations. *Structural and Multidisciplinary Optimization 32*, 5 (2006), 369–382.

[28] Jenatton, R., Archambeau, C., González, J., and Seeger, M. Bayesian optimization with tree-structured dependencies. In *International Conference on Machine Learning* (2017), PMLR, pp. 1655–1664.

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

[30] Kandasamy, K., Schneider, J., and Póczos, B. High dimensional bayesian optimisation and bandits via additive models. In *International conference on machine learning* (2015), PMLR, pp. 295–304.

[31] Keane, A. J. Statistical improvement criteria for use in multiobjective design optimization. *AIAA journal 44*, 4 (2006), 879–891.

[32] Klein, A., Falkner, S., Bartels, S., Hennig, P., and Hutter, F. Fast bayesian optimization of machine learning hyperparameters on large datasets. In *Artificial Intelligence and Statistics* (2017), PMLR, pp. 528–536.

[33] Knowles, J. Parego: A hybrid algorithm with on-line landscape approximation for expensive multiobjective optimization problems. *IEEE Transactions on Evolutionary Computation 10*, 1 (2006), 50–66.

[34] Kuleshov, V., and Precup, D. Algorithms for multi-armed bandit problems. *arXiv preprint arXiv:1402.6028* (2014).

[35] Kurucz, M., Benczúr, A. A., and Csalogány, K. Methods for large scale svd with missing values. In *Proceedings of KDD cup and workshop* (2007), vol. 12, Citeseer, pp. 31–38.

[36] Kushner, H. J. A new method of locating the maximum point of an arbitrary multipeak curve in the presence of noise. In *Joint Automatic Control Conference* (1963), vol. 1, pp. 69–79.

[37] Lee, E. H., Perrone, V., Archambeau, C., and Seeger, M. Cost-aware bayesian optimization. *arXiv preprint arXiv:2003.10870* (2020).

[38] Li, L., Jamieson, K., DeSalvo, G., Rostamizadeh, A., and Talwalkar, A. Hyperband: A novel bandit-based approach to hyperparameter optimization. *The Journal of Machine Learning Research 18*, 1 (2017), 6765–6816.

[39] Little, R. J., and Rubin, D. B. *Statistical analysis with missing data*, vol. 793. John Wiley & Sons, 2019.

[40] Luong, P. Bayesian optimization with missing inputs. `https://github.com/huuphuc2609/BOMI`, 2020. Online; accessed 14.01.2022.

[41] Luong, P., Nguyen, D., Gupta, S., Rana, S., and Venkatesh, S. Adaptive cost-aware bayesian optimization. *Knowledge-Based Systems 232* (2021), 107481.

[42] Luong, P., Nguyen, D., Gupta, S., Rana, S., and Venkatesh, S. Bayesian optimization with missing inputs. In *ECML PKDD 2020:*

Joint European Conference on Machine Learning and Knowledge Discovery in Databases (2021), Springer, pp. 691–706.

[43] Minami, T., Kawata, M., Fujita, T., Murofushi, K., Uchida, H., Omori, K., and Okuno, Y. Prediction of repeat unit of optimal polymer by bayesian optimization. *MRS Advances 4*, 19 (2019), 1125–1130.

[44] Mnih, A., and Salakhutdinov, R. R. Probabilistic matrix factorization. In *Advances in neural information processing systems* (2008), pp. 1257–1264.

[45] Mockus, J. *Bayesian approach to global optimization: theory and applications*, vol. 37. Springer Science & Business Media, 2012.

[46] Mockus, J., and Mockus, L. Bayesian approach to global optimization and application to multiobjective and constrained problems. *Journal of optimization theory and applications 70*, 1 (1991), 157–172.

[47] Mockus, J., Tiesis, V., and Zilinskas, A. The application of bayesian methods for seeking the extremum. *Towards global optimization 2*, 117-129 (1978), 2.

[48] Neal, R. M., et al. Mcmc using hamiltonian dynamics. *Handbook of markov chain monte carlo 2*, 11 (2011), 2.

[49] Neumann-Brosig, M., Marco, A., Schwarzmann, D., and Trimpe, S. Data-efficient autotuning with bayesian optimization: An industrial control study. *IEEE Transactions on Control Systems Technology 28*, 3 (2019), 730–740.

[50] Nikitin, A., Fastovets, I., Shadrin, D., Pukalchik, M., and Oseledets, I. Bayesian optimization for seed germination. *Plant methods 15*, 1 (2019), 43.

[51] Pacchiardi, L. Python implementation of bayesian probabilistic matrix factorization. `https://github.com/LoryPack/BPMF`, 2018. Online; accessed 14.01.2022.

[52] pandas development team, T. pandas-dev/pandas: Pandas 1.1.3, Oct. 2020.

[53] Paterek, A. Improving regularized singular value decomposition for collaborative filtering. In *Proceedings of KDD cup and workshop* (2007), vol. 2007, pp. 5–8.

[54] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research 12* (2011), 2825–2830.

[55] Rasmussen, C. E. Gaussian processes in machine learning. In *Summer school on machine learning* (2003), Springer, pp. 63–71.

[56] Robbins, H. Some aspects of the sequential design of experiments. *Bulletin of the American Mathematical Society 58*, 5 (1952), 527–535.

[57] Russo, D., Van Roy, B., Kazerouni, A., Osband, I., and Wen, Z. A tutorial on thompson sampling. *arXiv preprint arXiv:1707.02038* (2017).

[58] Saikai, Y., Mitchell, P. D., et al. Bayesian optimization for precision agriculture. In *2019 Annual Meeting, July 21-23, Atlanta, Georgia* (2019), Agricultural and Applied Economics Association.

[59] Salakhutdinov, R., and Mnih, A. Bayesian probabilistic matrix factorization using markov chain monte carlo. In *Proceedings of the 25th international conference on Machine learning* (2008), pp. 880–887.

[60] Schmidt, M. N., Winther, O., and Hansen, L. K. Bayesian non-negative matrix factorization. In *International Conference on Independent Component Analysis and Signal Separation* (2009), Springer, pp. 540–547.

[61] Slivkins, A. Introduction to multi-armed bandits. *arXiv preprint arXiv:1904.07272* (2019).

[62] Śmieja, M., Struski, Ł., Tabor, J., and Marzec, M. Generalized rbf kernel for incomplete data. *Knowledge-Based Systems 173* (2019), 150–162.

[63] Smola, A. J., Vishwanathan, S., and Hofmann, T. Kernel methods for missing variables. In *International Workshop on Artificial Intelligence and Statistics* (2005), PMLR, pp. 325–332.

[64] Snoek, J., Larochelle, H., and Adams, R. P. Practical bayesian optimization of machine learning algorithms. *Advances in neural information processing systems 25* (2012).

[65] Sóbester, A., Leary, S. J., and Keane, A. J. A parallel updating scheme for approximating and optimizing high fidelity computer simulations. *Structural and multidisciplinary optimization 27*, 5 (2004), 371–383.

[66] Srebro, N., and Jaakkola, T. Weighted low-rank approximations. In *Proceedings of the 20th International Conference on Machine Learning (ICML-03)* (2003), pp. 720–727.

[67] Srinivas, N., Krause, A., Kakade, S. M., and 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.

[68] Sutton, R. S., and Barto, A. G. *Reinforcement learning: An introduction*. MIT press, 2018.

[69] Tanaka, R., and Iwata, H. Bayesian optimization for genomic selection: a method for discovering the best genotype among a large number of candidates. *Theoretical and applied genetics 131*, 1 (2018), 93–105.

[70] Thathachar, M. A class of rapidly converging algorithms for learning automata. In *IEEE International Conference on Cybernetics and Society* (1984), pp. 602–606.

[71] Tsai, S.-F., Shen, C.-C., and Liao, C.-T. Bayesian optimization approaches for identifying the best genotype from a candidate population. *Journal of Agricultural, Biological and Environmental Statistics* (2021), 1–19.

[72] Vahid, A., Rana, S., Gupta, S., Vellanki, P., Venkatesh, S., and Dorin, T. New bayesian-optimization-based design of high-strength 7xxx-series alloys from recycled aluminum. *JOM 70*, 11 (2018), 2704–2709.

[73] Van Buuren, S. *Flexible imputation of missing data*. CRC press, 2018.

[74] van Hoof, J., and Vanschoren, J. Hyperboost: Hyperparameter optimization by gradient boosting surrogate models. *arXiv preprint arXiv:2101.02289* (2021).

[75] Virtanen, S., Klami, A., Khan, S., and Kaski, S. Bayesian group factor analysis. In *Artificial Intelligence and Statistics* (2012), PMLR, pp. 1269–1277.

[76] Wang, Z., Gehring, C., Kohli, P., and Jegelka, S. Batched large-scale bayesian optimization in high-dimensional spaces. In *International Conference on Artificial Intelligence and Statistics* (2018), PMLR, pp. 745–754.

[77] Wang, Z., Hutter, F., Zoghi, M., Matheson, D., and de Feitas, N. Bayesian optimization in a billion dimensions via random embeddings. *Journal of Artificial Intelligence Research 55* (2016), 361–387.

[78] Williams, C. K., and Rasmussen, C. E. *Gaussian processes for machine learning*, vol. 2. MIT press Cambridge, MA, 2006.

[79] Yamaguchi, K., Phenisee, S. E., Chen, Z., Salviato, M., and Yang, J. Ply-drop design of non-conventional laminated composites using bayesian optimization, 2020.

[80] Zhang, X., Lu, Y., Xiao, H., and Peterlik, H. Effect of hot stretching graphitization on the structure and mechanical properties of rayon-based carbon fibers. *Journal of Materials Science 49*, 2 (2014), 673–684.

[81] Zhilinskas, A. Single-step bayesian search method for an extremum of functions of a single variable. *Cybernetics 11*, 1 (1975), 160–166.