Automatic Construction of Nonparametric Relational Regression Models for Multiple Time Series

Yunseong Hwang[†1] YUNSEONG.HWANG@NAVERCORP.COM
Anh Tong[†] ANHTH@UNIST.AC.KR
Jaesik Choi[†] JAESIK@UNIST.AC.KR
[†]Ulsan National Institute of Science and Technology, Ulsan, 44919, Korea

Abstract

Gaussian Processes (GPs) provide a general and analytically tractable way of modeling complex time-varying, nonparametric functions. The Automatic Bayesian Covariance Discovery (ABCD) system constructs natural-language description of time-series data by treating unknown time-series data nonparametrically using GP with a composite covariance kernel function. Unfortunately, learning a composite covariance kernel with a single time-series data set often results in less informative kernel that may not give qualitative, distinctive descriptions of data. We address this challenge by proposing two relational kernel learning methods which can model multiple time-series data sets by finding common, shared causes of changes. We show that the relational kernel learning methods find more accurate models for regression problems on several real-world data sets; US stock data, US house price index data and currency exchange rate data.

1. Introduction

Gaussian Processes (GPs) provide a general and analytically tractable way of capturing complex time-varying, nonparametric functions. The time varying parameters of GPs can be explained as a composition of base kernels such as linearity, smoothness or periodicity in that covariance kernels are closed under addition and multiplication. The Automatic Bayesian Covariance Discovery (ABCD) system (Lloyd et al., 2014) constructs natural-language description of time-series data by treating unknown time-series data nonparametrically using GPs.

It is important to find data dependencies and structure in time-series data. GPs represent data in a non-parametric way with a mean function and a covariance kernel function. The covariance kernel function determines correlation patterns between the data points. Therefore, learning a proper kernel is essential to model data points with GPs (Rasmussen & Williams, 2006; Diosan et al., 2007; Bing et al., 2010; Klenske et al., 2013; Lloyd, 2014). Unfortunately, finding an appropriate kernel often requires manual encoding by human experts or reduction to a simple problem estimating parameters of a fixed, predefined kernel structure.

The covariance kernels of GPs are known to be closed under addition and multiplication (Duvenaud et al., 2013). Thus, a sequence of complex real-valued variables could be explained by a compositional kernel with base kernels and kernel operations (Bach et al., 2004; Candela & Rasmussen, 2005; Wilson & Adams, 2013). Recently, kernel operation grammars and a framework for an automatic discovery of a compositional kernel have been proposed (Lloyd et al., 2014). This framework is flexible and interpretable in that the framework automatically discovers a complex composition of interpretable base kernels. Once a compositional kernel is found, the individual base components can be construed in a human readable form. This capability for decomposition into multiple components and for interpretability, shed light on finding shared structure given multiple sets of data.

Finding a shared structure in multiple sequences may reveal the common covariance structures of the sequences beyond the patterns in a single sequence (Chu et al., 2006; Xu et al., 2009; Wilson & Ghahramani, 2011). As a typical example in economics, the multivariate view is central where each variable is normally viewed in the context of relationships to other variables, as with exchange rates affecting gross domestic product (GDP).

We propose two Relational Multi-Kernel Learning methods (explained in Section 3) to find a shared covariance ker-

[1]Currently at NAVER Corp., Seongnam, 13561, Korea

Proceedings of the 33^{rd} International Conference on Machine Learning, New York, NY, USA, 2016. JMLR: W&CP volume 48.

nel for multiple sets of data sequences. Our algorithm discovers both a shared composite kernel, which explains the common causes of changes in multiple data sets, and individual components (scale factors and distinctive kernels), which explain changes in individual data sets. Since GPs with the learned kernel are non-parametric, we can represent any data which is known to have the same relationship with the training data.

This paper is organized as follows. Section 2 introduces Gaussian Process (GP) and the Automatic Bayesian Covariance Discovery (ABCD) system. Section 3 presents our main contribution and algorithm, Relational Automatic Bayesian Covariance Discovery (RABCD). Section 4 discusses related work. Section 5 shows experimental results with real-world data sets followed by conclusions in Section 6.

2. Automatic Bayesian Covariance Discovery

Here, we briefly explain Gaussian Processes (GPs) and then introduce the Automatic Bayesian Covariance Discovery (ABCD) system (Lloyd et al., 2014), which is an extension of Compositional Kernel Learning (CKL) (Duvenaud et al., 2013).

2.1. Gaussian Process

GPs are distributions over functions such that any finite set of function evaluations, $(f(x_1), f(x_2), \ldots, f(x_N))$ form a multivariate Gaussian distribution (Rasmussen & Williams, 2006). As a multivariate Gaussian distribution is specified by its mean vector μ and covariance matrix $\mathbf{\Sigma}$, a GP is specified by its mean function, $\mu(x)$=$\mathbb{E}(f(x))$ and covariance kernel function, $k(x, x')$=$\mathrm{Cov}(f(x), f(x'))$. Evaluations of the two functions on a finite set of points correspond to the mean vector and the covariance matrix for the multivariate Gaussian distribution, like μ_i=$\mu(x_i)$ and $\mathbf{\Sigma}_{ij}$=$k(x_i, x_j)$. When a GP is specified, we can calculate the marginal likelihood of given data or can derive predictive distribution for new points given existing data.

Throughout this paper, we will use the following notations for GPs. If a function or evaluations of the function f are drawn from a GP specified by its mean function $\mu(x)$ and covariance kernel function $k(x, x')$,

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

We may occasionally omit the arguments of the functions, x and x', for simplicity and just say k is a kernel function. For zero-mean GPs, we put 0 in the place of mean function, which means $\mu(x)$=0. The covariance kernel functions often have its hyperparameters which are free parameters in defining a kernel function. For example, the squared exponential kernel function $k(x, x') = \sigma^2 \exp(|x - x'|^2/\ell)$ has two hyperparameters, σ and ℓ. In the following, we will use notation $k(x, x'; \theta)$ as a kernel function that has a vector θ as its hyperparameters.

2.2. Compositional Kernel Learning

Compositional Kernel Learning (CKL) constructs and finds richer kernels which are composed of several base kernels and operations. In theory, any positive definite kernels are closed under addition and multiplication (Duvenaud et al., 2013). Here, each base kernel expresses each distinctive feature such as linearity or periodicity. The kernel operations include not only addition and multiplication but also the so-called change-point and change-window operation to deal with abrupt structural changes.

2.2.1. Base Kernels

Five base kernels are used for making compositional kernels. Each kernel encodes different characteristics of functions, which further enables the generalization of structure and inference with new data.

Base Kernels	Encoding Function
White Noise (WN)	Uncorrelated noise
Constant (C)	Constant functions
Linear (LIN)	Linear functions
Squared Exponential (SE)	Smooth functions
Periodic (PER)	Periodic functions

2.2.2. Operations

The first operation is addition which sums multiple kernel functions and makes a new kernel function.

$$k'(x, x') = k_1(x, x') + k_2(x, x').$$

This operation works as a superposition of multiple independent covariance functions. In general, if $f_1 \sim \mathcal{GP}(\mu_1, k_1), f_2 \sim \mathcal{GP}(\mu_2, k_2)$ then $f := f_1 + f_2 \sim \mathcal{GP}(\mu_1 + \mu_2, k_1 + k_2)$.

ABCD assumes zero mean for GPs, since marginalizing over an unknown mean function can be equivalently expressed as a zero-mean GP with a new kernel (Lloyd et al., 2014). Under this assumption, the following multiplication operation is also applicable,

$$k'(x, x') = k_1(x, x') \times k_2(x, x').$$

In general, if $f_1 \sim \mathcal{GP}(0, k_1), f_2 \sim \mathcal{GP}(0, k_2)$ then $f := f_1 \times f_2 \sim \mathcal{GP}(0, k_1 \times k_2)$.

The third operation is the change-point (CP) operation. Given two kernel functions, k_1 and k_2, the new kernel function is represented as follows:

$$\begin{aligned} k'(x, x') = &\sigma(x)k_1(x, x')\sigma(x') \\ &+ (1 - \sigma(x))k_2(x, x')(1 - \sigma(x')) \end{aligned}$$

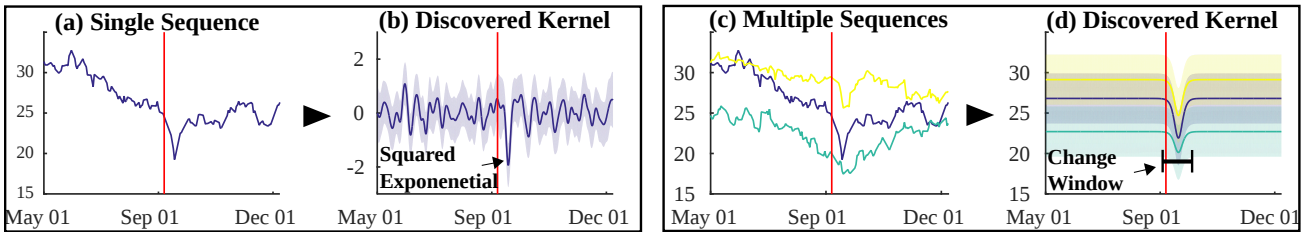


Figure 1: This figure shows components learned from CKL (left) and RKL (right). (a) and (b) represent adjusted closes of GE and learned components by CKL, respectively. (c) and (d) represent adjusted closes of 3 selected stocks (from top, yellow:XOM, purple:GE, green:MSFT) and learned (common) components by RKL. The red vertical lines indicate the September 11 attacks which occurred in 2001. RKL finds and explains the sudden drop after 911 by selectively applying a constant kernel around that period. However, CKL tries to fit that drop using a (less informative) rapidly varying squared exponential (SE) kernel function. We provide all components learned by CKL and RKL in the appendix.

where $\sigma(x)$ is a sigmoidal function which lies between 0 and 1, and ℓ is the change-point. The change-point operation divides function domain (i.e., time) into two sides and applies a different kernel function on each side. To generalize, if $f_1 \sim \mathcal{GP}(\mu_1, k_1), f_2 \sim \mathcal{GP}(\mu_2, k_2)$ then $f := \sigma(x)f_1 + (1-\sigma(x))f_2 \sim \mathcal{GP}(\sigma(x)\mu_1 + (1-\sigma(x))\mu_2, \sigma(x)k_1\sigma(x') + (1-\sigma(x))k_2(1-\sigma(x')))$.

Finally, the change-window (CW) operation applies the CP operation twice with two different change points ℓ_1 and ℓ_2. Given two sigmoidal functions $\sigma_1(x;\ell_1)$ and $\sigma_2(x;\ell_2)$ where $\ell_1 < \ell_2$, the new function will be $f := \sigma_1(x)f_1(1-\sigma_2(x)) + (1-\sigma_1(x))f_2\sigma_2(x)$, which applies the function f_1 to the window (ℓ_1, ℓ_2). A composite kernel expression after the change-window operation will be as follows:

$$k'(x,x') = \sigma_1(x)(1-\sigma_2(x))k_1(x,x')\sigma_1(x')(1-\sigma_2(x')) + (1-\sigma_1(x))\sigma_2(x)k_2(x,x')(1-\sigma_1(x'))\sigma_2(x').$$

2.2.3. Search Grammar

ABCD searches a composite kernel based on the search grammar. The search grammar specifies how to develop the current kernel expression by applying the operations with the base kernels. The following rules are examples of typical search grammar:

$$\mathcal{S} \to \mathcal{S} + \mathcal{B} \qquad \mathcal{S} \to \mathcal{S} \times \mathcal{B}$$
$$\mathcal{S} \to \mathrm{CP}(\mathcal{S}, \mathcal{S}) \qquad \mathcal{S} \to \mathrm{CW}(\mathcal{S}, \mathcal{S})$$
$$\mathcal{S} \to \mathcal{B} \qquad \mathcal{S} \to \mathrm{C}$$

where $\mathcal{S}$ represents any kernel subexpression, $\mathcal{B}$ and $\mathcal{B}'$ are base kernels. For example, supposed that there is a kernel expression $\mathcal{E} = \mathcal{K}_1 + \mathcal{K}_2 + \mathcal{K}_3$ where $\mathcal{K}_i$ is a kernel expression which cannot be separated into summands. Then, kernel subexpression $\mathcal{S}$ can be the summation of a non-empty subset of $\mathcal{K}_i$. If we apply multiplication grammar on a subset $(\mathcal{K}_1 + \mathcal{K}_3)$ with a base kernel $\mathcal{B}_1$, the expanded kernel expression is $\mathcal{E}' = \mathcal{K}_2 + (\mathcal{K}_1 + \mathcal{K}_3) \times \mathcal{B}_1$

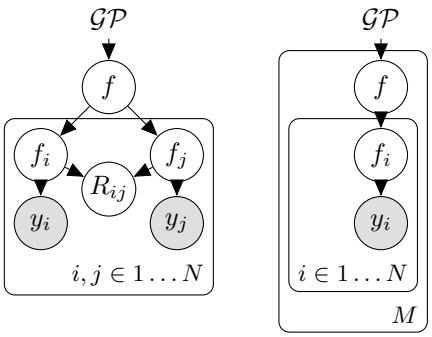
2.2.4. Algorithm

Given data and a maximum search depth, the algorithm gives a compositional kernel $k(x,x';\theta)$. Starting from the WN kernel, the algorithm expands the kernel expression based on the search grammar, optimizes hyperparameters for the expanded kernels, evaluates those kernels given the data and selects the best one among them. This procedure repeats. The next iteration in the procedure starts with the best composite kernel selected in the previous iteration. The conjugate gradient method is used when optimizing hyperparameters. Bayesian Information Criterion (BIC) (Schwarz, 1978) is used for the model evaluation. The BIC of model $\mathcal{M}$ with $|\mathcal{M}|$ number of free parameters and data $\mathcal{D}$ with $|\mathcal{D}|$ number of data points is:

$$\mathrm{BIC}(\mathcal{M}) = -2\log p(\mathcal{D}|\mathcal{M}) + |\mathcal{M}|\log|\mathcal{D}|. \quad (1)$$

The iteration continues until the specified maximum search depth is reached. During the iteration, the algorithm keeps the best model for the output.

Here is an example of how this algorithm works. Suppose that we start from the WN kernel. Then, we apply operators with some base kernels as described in the expansion grammar, such as WN $\to$ WN + SE, WN $\to$ WN + LIN and so on. With those expanded kernels, the algorithm optimizes hyperparameters of each expanded kernel. Now we have all the optimized kernels. Finally we compare the optimized ones and select the best kernel in terms of the BIC. This procedure repeats until we meet a certain depth of search. As an example, suppose that the best kernel selected in the previous step was WN + SE. Then we apply kernel expansion again like WN + SE $\to$ WN + SE $\times$ LIN and so on. Then we optimize the hyperparameters, select the best kernel, and then proceed to the next step.



(a) RGPs (b) RKL

Figure 2: Graphical representation of (a) RGPs (Chu et al., 2006) and (b) RKL. Here f denotes latent function. The subscripts of f_i and f_j denote sampled latent values for each point, as in $f_i = f(x_i)$. y_i and y_j are observed values for each point. N is the number of data points for each data set. M is the number of different data sets.

3. Relational Multi-Kernel Learning

Relational Kernel Learning (RKL) combines the ABCD system with Statistical Relational Learning (SRL). Given multiple sets of time-series data, RKL finds a composite kernel for GPs, which can describe the shared structure of the sets. We first introduce the RKL model with only a shared kernel, then presents a semi-RKL model with shared and individual kernels.

3.1. Relational Kernel Learning

RKL aims to find a model $\mathcal{M}$ that explains M multiple data sets, $\mathcal{D} = \{d_1, d_2, \dots, d_M\}$ well. To do so, we find a model that maximizes the likelihood, $p(\mathcal{D}|\mathcal{M}) = p(d_1, d_2, \dots, d_M|\mathcal{M})$. Here, $\mathcal{M}$ represents GP models. We assume the conditional independence of the likelihoods of each time-series data d_i. Hence, the log likelihood of the whole data is the sum of the log likelihoods of individual time-series:

$$\begin{aligned} \log p(d_1, \dots, d_M|\mathcal{M}) &= \log \prod\nolimits_i p(d_i|\mathcal{M}) \\ &= \sum\nolimits_i \log p(d_i|\mathcal{M}) \end{aligned} \tag{2}$$

This notation leads us to consider each data set separately and utilize the existing optimization technique and model evaluation form used for a single data set.

We assume that there is a single, shared factor that determines the covariance pattern for multiple data sets within the same domain, e.g., stock values of companies in a certain industrial sector. This acts as tying different factors that determine the covariance pattern of each data set as one single factor. By finding the factor, we achieve the shared compositional covariance kernel. Finding the shared factor in GPs is reduced to finding a shared covariance kernel function.

3.1.1. Description

Defining a GP requires mean function $\mu(x)$ and covariance function $k(x, x')$. The mean function is set to be a constant function which gives 0. The covariance function is divided into two parts, a functional part k, and its parameters θ. Here we say the functional part k is the function structure but the values of its hyperparameters are not specified yet. k is from a context-free language set yielded from context-free grammar G, which defines the search method while expanding the search tree in the kernel search algorithm. Here G contains base kernels like WN and operators like $+$ and $\times$. After functional part k is decided, the hyperparameter vector θ is decided. In addition, we have parameter vector σ where its length is twice the number of datasets. Each b_j (additive scaling factor) and v_j (multiplicative scaling factor) in $\sigma = [b_1, v_1, \dots, b_M, v_M]$ is added and multiplied to the kernel function $k(x, x'; \theta)$ before defining a GP prior for each data set, i.e. $k' = b_j^2 + v_j^2 \times k(x, x'; \theta)$ for each jth data set. These parameters normalize of scale variance to deal with the data sets with different scales of bias and variance in target value y. Summarizing all the process gives the following model.

$$\begin{aligned} k &\leftarrow G, \qquad \mathcal{GP} \leftarrow k, \theta, \sigma \\ f &\leftarrow \mathcal{GP}, \qquad \mathbf{y} \leftarrow f, \mathbf{X} \end{aligned} \tag{3}$$

Here f is the latent function from GP prior. $\mathbf{y}$ and $\mathbf{X}$ are $N \times 1$ and $N \times D$ matrices where N is the number of data points for each data set and D is the input dimension. For y_i which is the ith element of column vector $\mathbf{y}$, and $\mathbf{x}_i$ which is the ith row vector of matrix $\mathbf{X}$, $y_i = f(\mathbf{x}_i)$ holds.

Figure 3 shows the overall structure of the model. The GP prior is specified for each data set, from 1 to M. A mean function (which is zero here) and a kernel function specify a GP prior. The covariance kernel function is specified with the kernel structure and hyperparameters. Additive and multiplicative scaling factors, b_j and v_j for each data set, are added and multiplied to the kernel function and specifies a GP prior. The GP prior for each data set gives latent function f. From that latent function, we get latent values f_i for each data point. Finally y_i is generated from the latent value f_i. So for each data set, we have the following distribution:

$$K_{i,j} = b^2 + v^2 \times k(\mathbf{x}_i, \mathbf{x}_j; \theta) \tag{4}$$

$$\mathbf{y} \sim \mathcal{N}(0, K) \tag{5}$$

Here $\mathcal{N}(0, K)$ denotes a multivariate normal distribution with 0 mean vector and covariance matrix K.

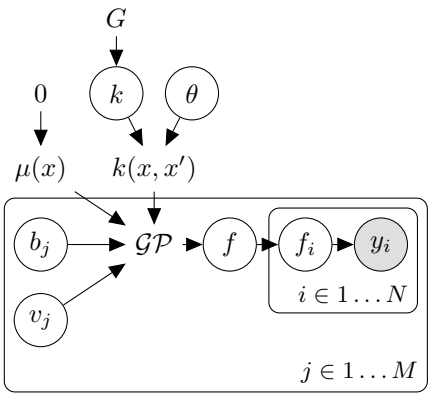


Figure 3: Graphical representation of RKL model with more steps in generating GP prior. k defines the kernel function's structure but not the values of its hyperparameters. θ completes the function by deciding the function's hyperparameter values. b_j and v_j are added and multiplied to $k(x, x')$ for each GP prior. This works as normalization of shift and scale variance of each data set.

3.1.2. Learning

Algorithm 1 presents a learning procedure to find an RKL model. This algorithm finds a single covariance kernel, shared by multiple data sets. To explain each line in detail, line 1 limits the depth of the search tree. Line 2 expands the given kernel k to multiple candidate kernels based on the expansion grammar G giving the set of expanded kernels K. For each expanded kernel, in line 4, thealgorithm optimizes hyperparameters and scaling factors of the kernel, that minimizes negative log-likelihood on the whole multiple data sets $\mathcal{D}$. In the optimization process, a conjugate gradient descent algorithm is used. After optimizing all the parameters for every candidate kernels in line 6, the algorithm selects the best kernel among those candidates that minimizes BIC. At the end, in line 8, it returns the best kernel.

The main difference is that we calculate the negative log marginal likelihood of the whole data as a summation of negative log marginal likelihood of each data set. This is possible because our RKL model shares a covariance kernel function through multiple data sets. This does not specify any correlation of variables across the different data sets. In other words, if we construct a single covariance matrix for all the data, we get a covariance matrix that has a block diagonal form. Thus we can factorize the likelihood $p(\mathcal{D}|\mathcal{M})$, where $\mathcal{M}$ is a model and D is a set of data, into a product of individual likelihoods $\prod_{d\in\mathcal{D}} p(d|\mathcal{M})$ as those individual likelihoods are independent. As we are dealing with the log likelihood, this can be represented as $\sum_{d\in\mathcal{D}} \log p(d|\mathcal{M})$.

Algorithm 1 Relational Kernel Learning

Require: initial kernel k, initial hyperparameters θ, initial scaling factors σ, multiple data sets $\mathcal{D} = \{d_1, \ldots, d_M\}$, expansion grammar G, maximum depth of search s

1: **for** $i \in 0 \ldots s$ **do**
2: $K \leftarrow \text{expand}(k, G)$
3: **for** $k \in K$ **do**
4: $k(\theta, \sigma) \leftarrow \text{argmin}_{(\theta,\sigma)} - \log p(\mathcal{D}|k)$
5: **end for**
6: $k \leftarrow \text{argmin}_{k\in K} \text{BIC}(k, \mathcal{D})$
7: **end for**
8: **return** k

3.2. Semi-Relational Kernel Learning

The assumption made by the RKL model is rather too strong to accommodate variations of individual data sequences. To handle this issue, we propose Semi-Relational Kernel Learning which loosens RKL's constraint by considering two parts of a structure, including a shared structure and an individualized structure.

3.2.1. Description

SRKL aims to learn a set of kernels

$$\mathbf{K} = \{K_j = K_S + K_{d_j} | d_j \in \mathcal{D}, j = 1, \ldots, M\},$$

where K_S is the shared kernel among M sequences, K_{d_j} is the distinctive kernel for j-th sequence. When we describe time series as types of trees, the shared kernel represents the common shape of trunk shared among those trees. The distinctive kernel interprets the small branches and leaves.

Ideally, the search grammar can be performed to discover one shared kernel and M distinctive kernels. The search space explodes in term of complexity $\mathcal{O}(n^{M+1})$ where n is the number of possible kernels on each search grammar tree for every depth. To reduce this intensive search, the distinctive kernel for each time series does not follow the extensive search grammar but is fixed by using the spectral mixture (SM) kernel (Wilson & Adams, 2013)

$$k(\tau) = \sum_{q=1}^{Q} w_q \prod \exp\{-2\pi^2 \tau_p^2 v_q^{(p)}\} \cos(2\pi \tau_p \mu_q^{(p)}),$$

where Q is the number of components, $\tau = x - x'$ is a P dimensional vector. SM is chosen because its expressiveness and ability to approximate a wide range of covariance kernels.

3.2.2. Learning

Algorithm 2 elucidates the learning procedure of the SRKL model. The search grammar keeps playing its role as a

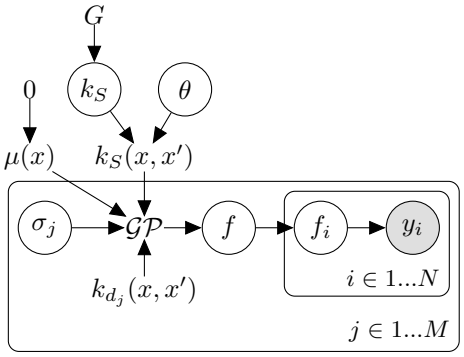


Figure 4: Graphical representation of SRKL model. Comparing to RKL, SRKL elaborates the GP kernel structure with an additional distinctive kernel for each data sequence.

Algorithm 2 Semi-Relational Kernel Learning

Require: data $\mathcal{D} = \{d_1, \ldots, d_M\}$, grammar G, maximum depth of search s

Initialize $\mathcal{K} \leftarrow \emptyset$
for $i \in 0 \ldots s$ **do**
  $K_S \leftarrow$ expand(G)
  $\Theta \leftarrow \emptyset$
  for $k_S \in K_S$ **do**
    Initialize $\theta^0 \leftarrow (\theta_S^0, \theta_1^0, \ldots, \theta_M^0, \sigma_1^0, \ldots, \sigma_M^0)$
    $k_j(\theta^0) \leftarrow k_S(\theta_S^0, \sigma_j^0) + k_{d_j}(\theta_j^0), j = 1 \ldots M$
    $\theta^* \leftarrow \text{argmin} \sum_{j=1}^{M} -\log p(\mathcal{D}|k_j(\theta))$
    $\Theta \leftarrow \Theta \cup \{(k_S, \theta^*)\}$
  end for
  $(\hat{k_S}, \hat{\theta}) \leftarrow \text{argmin}_{(k_S,\theta)\in\Theta}$ BIC $(k_S, \mathcal{D})$
  $\mathcal{K} \leftarrow \mathcal{K} \cup (\hat{k_S}, \hat{\theta}, \hat{\sigma})$
end for
return $\mathcal{K}$

generator of composite kernels for each depth s. For each kernel in the search space, the hyperparameters consist of shared hyperparamters θ_S, scale factors $\sigma_1, \ldots, \sigma_M$, distinctive hyperparameters $\theta_1, \ldots, \theta_M$. The optimal hyperparameters are learned based on finding the minimum negative log likelihood of data on kernels $\mathbf{K}$. The objective is to find the most common components between time series. The search grammar identifies the best shared kernel by the BIC score on a shared kernel K_S where the likelihood is computed by the summation of the likelihood of each time series with respect to the shared kernel.

During the learning procedure, there is a notable compromise between the shared kernel K_S and distinctive kernels K_{d_j}. At the beginning (lower levels of search grammar), the shared kernel is still coarse and not expressive enough. The distinctive kernel fills the gap between the true kernel and the shared kernel. When the search grammar gets deeper, K_S becomes more complex and makes K_{d_j} adapt

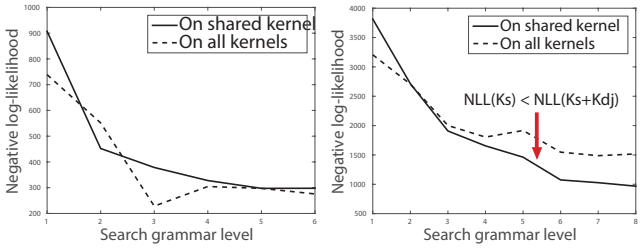


Figure 5: Negative log-likelihoods (NLL) made by K_S and K_j. (a) Non-overfitting case. The NLLs on K_j and K_S decrease together; (b) Overfitting starts from level 3 of search grammar. From level 3, the total NLL on K_S keeps going down while the total NLL on the whole kernel K_j starts increasing. K_{d_j} adds more complexity to the model and worsens the K_j.

to data not yet explained by K_S. Once K_S gets too complex to data, K_{d_j} will be unable to make improvement on K_j. The overfitting phenomena occurs when the negative log-likelihood made by K_j starts overwhelming the negative log-likelihood made by K_S as shown in Figure 5.

4. Related Work

4.1. Learning Composite GP Kernels

A composite GP kernel can greatly improves the performance of nonparametric regression models. Manually constructed composite models have shown to fit accurate GP models (Rasmussen & Williams, 2006; Klenske et al., 2013; Lloyd, 2014). Tree-shaped composite kernels have also been used with Support Vector Machine (SVM) (Diosan et al., 2007; Bing et al., 2010).

A weighted sum of base kernels is limited but also effective when building a composite kernel from base kernels. Multiple Kernel Learning (MKL) (Bach et al., 2004; Candela & Rasmussen, 2005; Wilson & Adams, 2013) find optimal weights in polynomial time when the component kernels and parameter are pre-specified.

Recently, an algorithm for learning a composite kernel (CKL) (Duvenaud et al., 2013), and a system for an automatic explanation of the learned composite kernel (ABCD) (Lloyd et al., 2014) have been proposed. Our RKL and SRKL are based on CKL and the ABCD system. However, our models expand the CKL for multiple time-series data and improves the interpretability of the individual time-series data by finding a shared relational structure.

4.2. Relational GPs for Continuous Data

Relational GPs can represent the relationships among multiple time-series data sets (Chu et al., 2006; Xu et al., 2009;

Wilson & Ghahramani, 2011). These models are based on pre-specified base kernels instead of composite kernel representations. A straightforward extension of relational GP with a composite kernel does not work unless the time-series data sets share the same hyperparameter. Our models are more flexible in handling relational data sets by introducing GPs with both shared and distinct, non-shared hyperparameters.

There is a large body of work attempting to represent the probabilistic knowledge formalisms in statistical relational learning (SRL) (Getoor & Taskar, 2007). SRL models for continuous variables (Wang & Domingos, 2008; Choi et al., 2010; Belle et al., 2015; Choi et al., 2015) and lifted inference algorithms have been proposed (Kimmig et al., 2015). Unfortunately, learning composite relational models from complex continuous data is still hard, and thus models are defined manually.

4.3. Multiple Output GPs

Multiple correlated data sets can be handled by multiple output GPs (Wei et al., 2007; Pan & Yao, 2008; Silva et al., 2008; Álvarez & Lawrence, 2011; Qiu et al., 2016). In principle, handling multiple output GPs is intractable because of the increased size of the full covariance function (matrix) (Álvarez & Lawrence, 2011). Thus, existing work exploits efficient ways to extract a common (but simple) structure given a fixed GP kernel. Our RKL and SRKL seek for a similar goal by finding a shared covariance kernel. However, our models with a shared composite GP kernel and distinctive components (scale factors and SM) are far more expressive.

5. Experimental Results

In this section, we compare two proposed models, RKL and SRKL, with CKL, the learning algorithm of the ABCD system.

5.1. Data sets

5.1.1. Stock Market

From US stock market data, we selected the 9 most valuable stocks, GE, MSFT, XOM, PFE, C, WMT, INTC, BP and AIG based on the market capitalization rankings as of 2001 (von Alten, 2001). The adjusted close of stock figures from 2001-05-29 to 2001-12-25 were collected from Yahoo finance (Yahoo Inc., 2015). Each stock's historical adjusted close in that period consists of 129 points. Thus, the total number of points is $1161 (= 129 \times 9)$. The collected period includes the September 11 attacks. After the 9/11 attacks, most of stock values show a steep drop and gradual recovery as time goes on. We arrange this data set into three different learning settings - STOCK3, STOCK6, and STOCK9. The suffix number indicates the number of most valuable stocks.

5.1.2. Housing Market

US house price index data are retrieved from S&P Dow Jones Indices (Guarino & Blitzer, 2015) as a seasonally adjusted home price index levels. We selected 6 cities: New York, Los Angeles, Chicago, Pheonix, San Diego and San Francisco based on the US city population rankings (United States Census Bureau, 2014) and where the house price index is available. The period is from the start of year 2004 to the end of year 2013 with monthly granularity, with a total of 120 points for each data set and 720 for all the data sets. There are smooth peaks around 2007 and drops until 2009 (the subprime mortgage crisis). Here, we have three learning settings - HOUSE2, HOUSE4, and HOUSE6 with top housing markets in terms of city population.

5.1.3. Emerging Currency Market

We collected US dollar to 4 currencies exchange rates from emerging markets; South African Rand (ZAR), Indonesian Rupiah (IDR), Malaysian Ringgit (MYR), and Russian Rouble (RUB). The currency data from 2015-06-28 to 2015-12-30 were acquired from Yahoo Finance (Yahoo Inc., 2015), containing 132 currency values for each currency. The financial market greatly fluctuated from the middle of September 2015 to the beginning of October 2015 as they were affected by several economic events including FED's announcement about policy changes in interest rates and falls in China's foreign exchange reserves. We call this data set as CURRENCY4.

5.2. Quantitative evaluations

Table 1 presents all experimental results for three different evaluation criteria; negative log-likelihood (NLL), and BIC on training data, and root mean square error on test (extrapolation) data.

5.2.1. Negative log likelihood and BIC

The first two groups of columns in Table 1 present how the models fit the data. RKL surpasses in most of data sets in term of the BIC. RKL model consistently keeps a small number of hyperparameters since it shares parameters through the data sets. With a single kernel and some scale factors, RKL express multiple sequences better than CKL in STOCK3, STOCK6, STOCK9, HOUSE4, and HOUSE6. It can be inferred that these time series are highly correlated.

SRKL, like RKL, also seeks for a general, shared kernel among time series data sets as RKL. However, the num-

Figure 6: A shared covariance kernel found by SRKL in the currency exchange rate data.

Table 1: Comparisons of CKL, RKL, and SRKL by NLL, BIC, and RMSE among on US stock data, US house price index data and current exchange rate data

	Negative log likelihood			Bayesian Information Criteria			Root mean square error		
Data set	CKL	RKL	SRKL	CKL	RKL	SRKL	CKL	RKL	SRKL
STOCK3	332.75	311.84	**304.05**	750.65	**665.09**	1251.62	0.40	0.78	**0.38**
STOCK6	**972.00**	1007.09	988.14	2219.71	**2066.18**	3333.21	3.69	5.75	**1.22**
STOCK9	1776.31	1763.96	**1757.11**	3985.03	**3626.00**	5633.33	8.35	9.77	**4.85**
HOUSE2	**264.69**	304.29	310.38	**634.00**	**634.76**	905.76	6.58	**2.75**	3.12
HOUSE4	594.79	**586.81**	1249.82	1424.18	**1221.88**	3326.94	5.84	3.66	**2.22**
HOUSE6	**849.64**	891.09	1495.40	2100.62	**1876.47**	4339.54	7.96	5.33	**3.10**
CURRENCY4	**578.35**	617.77	693.76	**1165.82**	1291.77	2269.17	330.00	282.24	**201.56**

ber of hyperparameters in the distinctive (spectral mixture) kernel contributes to SRKL's high BIC score. In addition, the SRKL kernels $K_j = K_S + K_{d_j}$ is more complex than other kernels. The complexity penalty $\log|K_j|/2$ (Rasmussen & Williams, 2006) in the negative log-likelihood term causes high negative log-likelihood.

5.2.2. Extrapolation performance

We assess performance using root mean square error (RMSE) between the predicted and actual values of the extrapolation after the end of the training period. For stock data, housing data, and currency exchange data, we respectively collected the next 14 days, 13 months, and 13 days of data.

The RMSEs of CKL, RKL, and SRKL are presented in the third group of columns in Table 1. Although SRKL has been seen possessing bigger BIC scores, it outperforms on most of data sets. It conveys the general information among time series via the shared kernel. SM kernels are in charge of the distinctive information, and complement favorably the shared kernel for each time series. RKL also outruns CKL on the housing data and the currency exchange data.

5.3. Qualitative Comparisons

RKL can find distinct signal components those are dominant in multiple sequences better than CKL. While CKL learns a model based on only a single dataset, RKL can refer to multiple datasets and thus provides much evidence to help decide whether a certain signal is really distinct or dominant. As with the example of the US stock market data, when we learned model for each data set using CKL, the most of the learned models could not specify the drop after the 9/11 attacks as a single component but just considered the drop as a kind of normal drift included in the smooth changes of the signal as time flows. However, in the case of the RKL, the model could find the component that solely explains the drop by exactly specifying the time window of the sudden drop and recovery after 9/11, similar to the Figure 1.

In currency exchange rate data, SRKL also finds a qualitatively important compositional kernel shortly written as CW(SE + CW(WN + SE, WN), C). The second change-window kernel presents a time period from mid September 2015 to beginning October 2015. This reveals the big changes in financial markets influenced by fiscal events, namely the FED's announcement about policy changes in interest rates (see Figure 6). ABCD captures a change-point on only one currency for the Indonesian Rupiah. The other results form ABCD do not show this change.

6. Conclusion

We proposed a nonparametric Bayesian framework that finds shared structure throughout multiple sets of data. The resulting Relational Kernel Learning (RKL) method provides a way of finding a shared kernel function that can describe multiple data with better BIC. We applied this model to several real world data, including US stock data, US house price index data and currency exchange rate data, to validate our approach.

Acknowledgments

This work is supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) grant funded by the Ministry of Science, ICT & Future Planning (MSIP) (NRF- 2014R1A1A1002662) and the NRF grant funded by the MSIP (NRF-2014M2A8A2074096).

References

Álvarez, Mauricio A. and Lawrence, Neil D. Computationally efficient convolved multiple output gaussian processes. *Journal of Machine Learning Research*, 12: 1459–1500, 2011.

Bach, Francis R., Lanckriet, Gert R. G., and Jordan, Michael I. Multiple kernel learning, conic duality, and the SMO algorithm. In *Proceedings of the Twenty-first International Conference on Machine Learning (ICML)*, 2004.

Belle, Vaishak, Passerini, Andrea, and Van den Broeck, Guy. Probabilistic inference in hybrid domains by weighted model integration. In *Proceedings of 24th International Joint Conference on Artificial Intelligence (IJCAI)*, 2015.

Bing, Wu, Wen-qiong, Zhang, Ling, Chen, and Jia-hong, Liang. A gp-based kernel construction and optimization method for rvm. In *Computer and Automation Engineering (ICCAE)*, volume 4, pp. 419–423, 2010.

Candela, Joaquin Quiñonero and Rasmussen, Carl Edward. A unifying view of sparse approximate gaussian process regression. *Journal of Machine Learning Research*, 6: 1939–1959, 2005.

Choi, Jaesik, Amir, Eyal, and Hill, David J. Lifted inference for relational continuous models. In *Proceedings of the Twenty-Sixth Conference on Uncertainty in Artificial Intelligence (UAI)*, pp. 126–134, 2010.

Choi, Jaesik, Amir, Eyal, Xu, Tianfang, and Valocchi, Albert J. Learning relational kalman filtering. In *Proceedings of the Twenty-Ninth Conference on Artificial Intelligence (AAAI)*, pp. 2539–2546, 2015.

Chu, Wei, Sindhwani, Vikas, Ghahramani, Zoubin, and Keerthi, S. Sathiya. Relational learning with gaussian processes. In *Proceedings of the Twentieth Annual Conference on Neural Information Processing Systems (NIPS)*, pp. 289–296, 2006.

Diosan, Laura, Rogozan, Alexandrina, and Pécuchet, Jean-Pierre. Evolving kernel functions for svms by genetic programming. In *The Sixth International Conference on Machine Learning and Applications (ICMLA)*, pp. 19–24, 2007.

Duvenaud, David K., Lloyd, James Robert, Grosse, Roger B., Tenenbaum, Joshua B., and Ghahramani, Zoubin. Structure discovery in nonparametric regression through compositional kernel search. In *Proceedings of the 30th International Conference on Machine Learning (ICML)*, pp. 1166–1174, 2013.

Getoor, Lise and Taskar, Ben. *Introduction to statistical relational learning*. MIT press, 2007.

Guarino, David and Blitzer, David. S&P/Case-Shiller home price tiered index levels. https://us.spindices.com/indices/real-estate/sp-case-shiller-20-city-composite-home-price-index, 2015. Accessed: 2015-09-15.

Kimmig, Angelika, Mihalkova, Lilyana, and Getoor, Lise. Lifted graphical models: a survey. *Machine Learning*, 99(1):1–45, 2015.

Klenske, Edgar D., Zeilinger, Melanie Nicole, Schölkopf, Bernhard, and Hennig, Philipp. Nonparametric dynamics estimation for time periodic systems. In *the 51st Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pp. 486–493, 2013.

Lloyd, James Robert. Gefcom2012 hierarchical load forecasting: Gradient boosting machines and gaussian processes. *International Journal of Forecasting*, 30(2):369–374, 2014.

Lloyd, James Robert, Duvenaud, David K., Grosse, Roger B., Tenenbaum, Joshua B., and Ghahramani, Zoubin. Automatic construction and natural-language description of nonparametric regression models. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence (AAAI)*, pp. 1242–1250, 2014.

Pan, Jiazhu and Yao, Qiwei. Modelling multiple time series via common factors. *Biometrika*, 95(2):365–379, 2008.

Qiu, Huitong, Han, Fang, Liu, Han, and Caffo, Brian. Joint estimation of multiple graphical models from high dimensional time series. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 78(2):487–504, 2016. ISSN 1467-9868.

Rasmussen, Carl Edward and Williams, Christopher K. I. *Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning)*. The MIT Press, 2006.

Schwarz, Gideon. Estimating the dimension of a model. *The annals of statistics*, 6(2):461–464, 1978.

Silva, Ricardo, Chu, Wei, and Ghahramani, Zoubin. Hidden common cause relations in relational learning. In *Proceedings of the Twentieth Annual Conference on Neural Information Processing Systems (NIPS)*, pp. 1345–1352, 2008.

United States Census Bureau. Population estimates: Historical data. https://www.census.gov/popest/data/historical/index.html, 2014. Accessed: 2015-09-15.

von Alten, Tom. Top 100 market capitalization. http://fortboise.org/top100mktcap.html, 2001. Accessed: 2015-09-15.

Wang, Jue and Domingos, Pedro M. Hybrid markov logic networks. In *Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence (AAAI)*, pp. 1106–1111, 2008.

Wei, Xing, Sun, Jimeng, and Wang, Xuerui. Dynamic mixture models for multiple time-series. In *Proceedings of the 20nd International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 2909–2914, 2007.

Wilson, Andrew Gordon and Adams, Ryan Prescott. Gaussian process kernels for pattern discovery and extrapolation. In *Proceedings of the 30th International Conference on Machine Learning (ICML)*, pp. 1067–1075, 2013.

Wilson, Andrew Gordon and Ghahramani, Zoubin. Generalised wishart processes. In *Proceedings of the Twenty-Seventh Conference on Uncertainty in Artificial Intelligence (UAI)*, pp. 736–744, 2011.

Xu, Zhao, Kersting, Kristian, and Tresp, Volker. Multi-relational learning with gaussian processes. In *Proceedings of the 21st International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 1309–1314, 2009.

Yahoo Inc. Yahoo finance - business finance, stock market, quotes, news. http://finance.yahoo.com/, 2015. Accessed: 2016-01-20.

A. Appendix

A.1. Components Extracted from the Stock Market Data

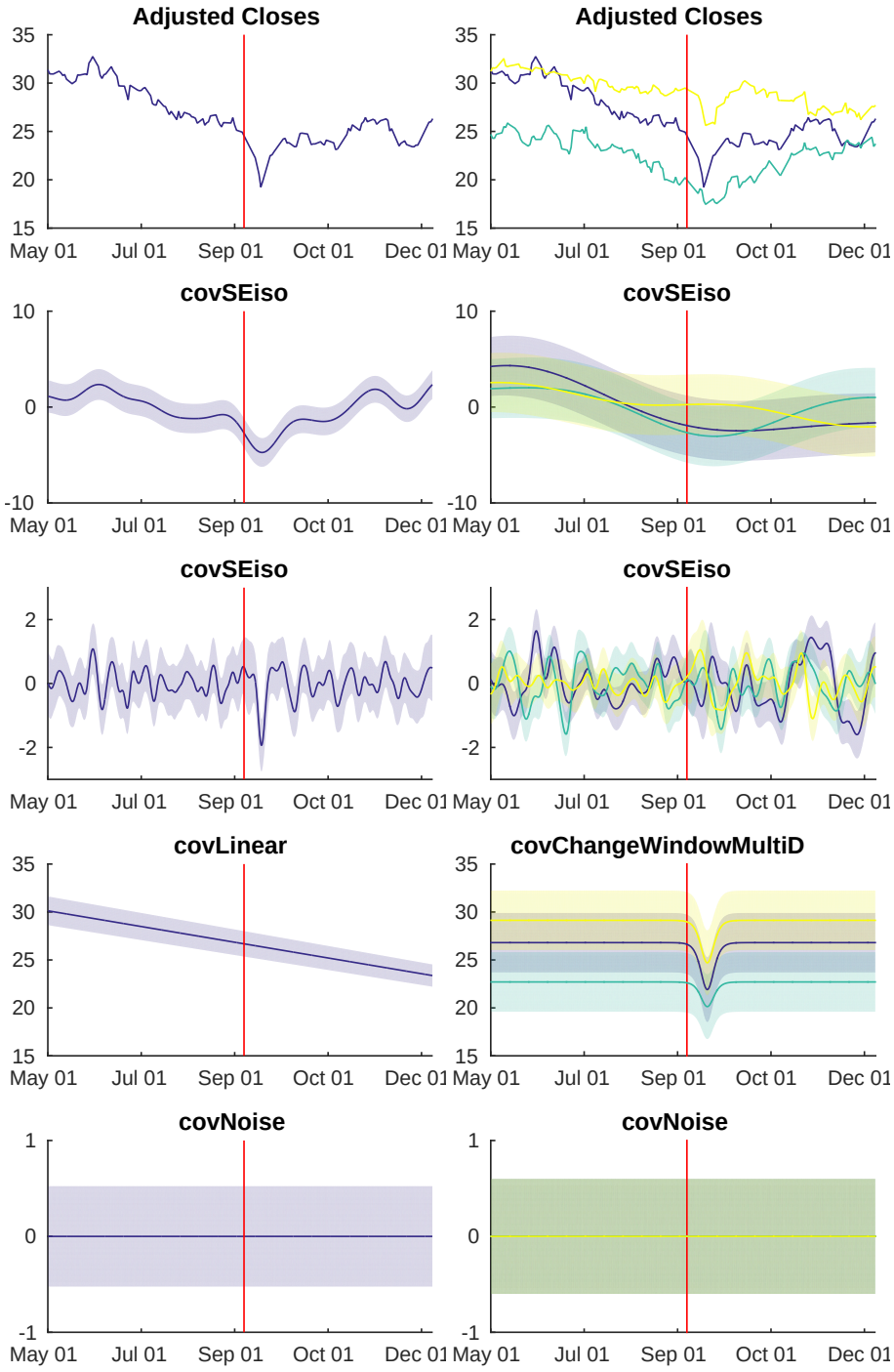


Figure 7: Figure that shows decomposition of adjusted closes of stock values. x axis is time and y axis is stock values. Red vertical line is at '2001/09/11' which is the day of 911. The third component of right side, 'covChangeWindowMultiD' captures sudden drop of stock values after the 911.

Time series	CKL	SRKL
GE	1.86	**0.48**
MSFT	3.60	**0.76**
XOM	1.18	**0.70**
PFE	1.63	**0.63**
C	1.10	**0.53**
WMT	1.84	**0.42**
INTC	1.22	**0.85**
BP	**1.01**	1.33
AIG	1.58	**0.94**

Table 2: Standardized RMSEs of stock data set

Time series	CKL	SRKL
New York	11.66	**9.97**
Los Angeles	2.75	**0.53**
Chicago	6.13	**1.24**
Pheonix	**3.39**	3.60
San Diego	7.94	**1.51**
San Francisco	2.65	**1.16**

Table 3: Standardized RMSEs of housing price data set

Time series	CKL	SRKL
IDR	3.55	**2.17**
MYR	**1.37**	2.35
ZAR	2.98	**1.32**
RUB	1.48	**1.29**

Table 4: Standardized RMSEs of currency exchange data set

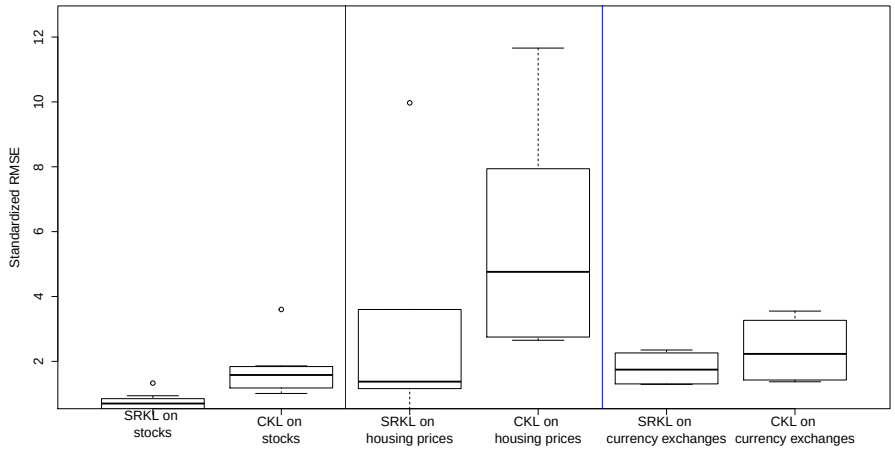


Figure 8: Box plot of standardized RMSEs on each data set

A.2. Details on standardized RMSEs

We provide the standardized RMSEs of all data sets in Table 2, Table 3, and Table 4. Moreover, Figure 8 shows the significant statistical improvement of SRKL in terms of extrapolation as discussed in the experiment section.

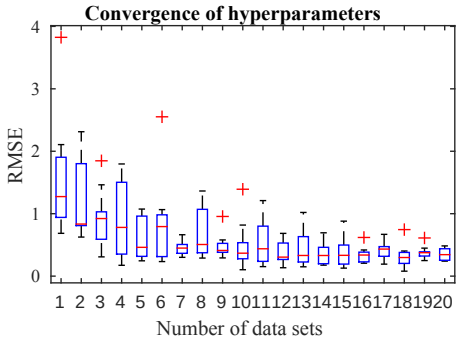


Figure 9: The convergence of error between the optimized hyperparameter and original true parameter that was used when generating data. The horizontal axis denotes the number of data sets used in hyperparameter optimization. The vertical axis shows the value of root mean squared error between optimized and true hyperparameters. Each boxplot shows the distribution of errors for each step.

Model-Stock		NLL	P	BIC
CKL	GE	116.36	7	266.75
	MSFT	111.17	6	251.51
	XOM	82.24	9	208.23
	PFE	62.91	12	184.14
	C	424.23	10	897.07
	WMT	149.96	5	324.23
	Total	**946.89**	49	2219.71
RKL-Total		1007.09	**18**	**2066.18**
SRKL-Total		988.15	204	3333.21

Table 5: The experimental comparisons of GP models with individual CKL, RKL and SRKL in the stock data set (top 6 US stocks in 2001). For each column, NLL means the negative log likelihood, P is the number of hyperparameters and BIC is the Bayesian information criterion. CKL is slightly better in a simple aggregation of NLL only. However, our RKL outperforms CKL in the BIC due to a significantly lower 21 parameters (P) compared to 49 parameters in CKL.

A.3. Learning Hyperparameters in Synthetic Data

First we had a covariance kernel function k that is already learned from the original ABCD algorithm. We generated 20 data sets from a GP prior with kernel k. Each data set has a different number of data points. After sampling the data, we tried to optimize the hyperparameters of k for $1, 2, \ldots, 20$ data sets chosen out of 20 sampled data sets, with the original hyperparameters as a starting point. We do this optimization 10 times for each number of data sets, from 1 to 20, with different combination of choice.

We calculated the root mean squared error (norm distance) between the true hyperparameter and the optimized hyperparameter vector. Figure 9 shows the result. The values along the horizontal axis means the number of data sets which are used in the optimization. Each boxplot shows the distribution of the error with 10 different results for each number of data sets used. It shows that as we increase the number of data sets in parameter optimization, the error between the optimized and the true hyperparameter decreases.

A.4. Experiment table with 6 stocks and 6 cities

Table 5 compares the fitness of the CKL, RKL and SRKL models. The results with line heading RKL are from our model. The others are from the original ABCD. The results with line heading Total is calculated by considering all of GP priors that are learned for each data set as a single model. So NLL and P values of individual models are added up and the total BIC of the CKL model as a single model was calculated using those values. In terms of negative log likelihood, applying ABCD individually gives better results. However if we compare the BIC, our model achieves better results, as our model has a reduced number of free parameters by sharing them through the data sets.

		RKL		SRKL		CKL	
Stocks	N	P	BIC	P	BIC	P	BIC
Top 3	387	**16**	**665.09**	108	1251.62	22	750.65
Top 6	774	**18**	**2066.18**	204	3333.21	49	2219.71
Top 9	1161	**32**	**3626.00**	300	5633.33	73	3985.03

Table 6: The BIC of CKL, RKL and SRKL in the stock data set. 'Top 3', 'Top 6' and 'Top 9' stocks were selected by their market capitalization ranks in 2011. As shown in Table 5, RKL requires fewer parameters than CKL. RKL models trained with 3 stocks and 6 stocks show better performance than individually optimized CKL models. When 9 stocks are considered, the individual CKL models show better performance than the single (shared) RKL model.

Model-City		NLL	P	BIC
CKL	New York	120.13	10	288.13
	Los Angeles	162.94	9	368.96
	Chicago	134.73	13	331.69
	Pheonix	101.76	11	256.17
	San Diego	155.64	12	368.73
	San Francisco	174.45	6	377.63
	Total	**849.64**	61	2100.62
RKL-Total		891.09	**33**	**1972.58**
SRKL-Total		1495.40	205	3707.94

Table 7: A comparison of BIC between individual CKL RKL and SRKL, with house data. For each column, NLL means negative log likelihood, P is the number of hyperparameters and BIC is the Bayesian information criterion. For each row, RKL is the result from our model. From New York to San Francisco, those results are from individual cities. Total is summation of those individual results.

Table 6 compares the fitness of the models with different numbers of data sets used in training. As the number of data sets increases, the number of parameters needed also increases. However RKL shows less need for parameters through the whole setup, since our model shares parameters though the data. RKL also performs well in terms of the BIC for TOP3 and TOP6. However as the number of data sets increases, the performance of the original CKL model surpasses our model, at TOP9. This is because our model fits the general structure but not individual specific ones. As the number of data sets increases, the learned structure cannot fully explain the individual specific patterns. And these accumulated errors in fitness, which is the NLL, offsets the advantages of the BIC which is from the reduced number of parameters.

Table 7 shows the fitness of models between our model and the original ABCD model. Similar to the stock market data case, the first line is our model and the others are from the original ABCD. Our model shows better results, a reduced number of free parameters and smaller BIC compared to Total case. However the original model still shows better fitness if we only consider NLL. From this experiment we can again confirm that the major contribution of the smaller BIC comes from the reduced number of parameters.

Table 8 compares results between our model and the ABCD for different numbers of data sets. Our model shows reduced numbers of parameters over all different sets of data since our model shares kernel parameters for multiple data sets. The

		RKL		SRKL		CKL	
SET	N	P	BIC	P	BIC	P	BIC
Top 2 cities	240	**11**	634.76	52	905.76	20	**634.00**
Top 4 cities	480	**18**	**1221.88**	134	3326.94	38	1424.18
Top 6 cities	720	**33**	**1876.47**	109	4339.54	61	2100.62

Table 8: The BIC of CKL, RKL and SRKL in the housing market data set. 'Top 2', 'Top 4' and 'Top 6' US cities were selected in terms of their city population rank. The BICs of the RKL models are similar or better than the BICs of individually trained CKL models.

original model shows better results in terms of BIC for the top 2 indices. However other than that, our model shows better results in terms of both number of parameters and BIC.