Stepwise Model Selection for Sequence Prediction via Deep Kernel Learning

Yao Zhang
University of Cambridge

Daniel Jarrett
University of Cambridge

Mihaela van der Schaar
University of Cambridge, UCLA
The Alan Turing Institute

Abstract

An essential problem in automated machine learning (AutoML) is that of model selection. A unique challenge in the sequential setting is the fact that the optimal model *itself* may vary over time, depending on the distribution of features and labels available up to each point in time. In this paper, we propose a novel Bayesian optimization (BO) algorithm to tackle the challenge of model selection in this setting. This is accomplished by treating the performance at each time step as its own black-box function. In order to solve the resulting multiple black-box function optimization problem *jointly* and *efficiently*, we exploit potential correlations among black-box functions using deep kernel learning (DKL). To the best of our knowledge, we are the first to formulate the problem of *stepwise* model selection (SMS) for sequence prediction, and to design and demonstrate an efficient joint-learning algorithm for this purpose. Using multiple real-world datasets, we verify that our proposed method outperforms both standard BO and multi-objective BO algorithms on a variety of sequence prediction tasks.

1 Introduction

Model selection is a central concern in automated machine learning (AutoML). Techniques using Bayesian optimization (BO) have proven popular and effective for this purpose Snoek et al. (2012), and have been extended to incorporate trade-offs between multiple objectives Hernández-Lobato et al. (2016); Picheny (2015); Shah and Ghahramani (2016), as well as accommodating transfer across multiple tasks Perrone et al. (2018); Swersky et al. (2013); Zhang et al. (2017). In this paper, we focus on Bayesian optimization for model selection in the *sequence prediction* setting—that is, where the underlying task is to emit predictions e_t at every step t given a sequence of observations $\{\mathbf{o}_t\}_{t=1}^{T}$ as input. Instead of studying the dynamics between or across different *tasks*, we concentrate on how the optimal model itself (for a fixed task) may vary *over time*, depending on the distribution of features and labels in the data available at each time.

Note that there are two different senses of changes "over time". The first concerns distribution shifts *across* successive batches of (static) data: If the data-generating process evolves across multiple datasets, existing models trained on prior data may require updating as new batches become available—that is, in order to continue to generalize well. This *sequential* process can be assisted for instance by hyperparameter transfer learning Perrone et al. (2018), and is not the focus of this work. The second type is more subtle, and is the motivation for our work: There may be temporal distribution shifts *within* the same dataset that we are attempting to learn from. Unlike in sequential hyperparameter transfer learning, here all the data we need is already available, and learning can in principle be done *jointly* across all time steps. Temporal distribution shifts often arise in healthcare, and can happen on both the individual and population level. As an example of the former, the risk factors for an adverse outcome at the beginning of a patient's hospital stay may generally be different from those that govern their condition towards the end of the episode. As an example of the latter, as a medical study progresses over time, the distribution of registered patients and their treatments and outcomes may undergo a shift. As noted in Oh et al. (2019); Wiens et al. (2016), while such phenomena are common in the medical setting, they are rarely addressed explicitly by current *single-model* techniques—potentially giving rise to suboptimal prediction performance.

In this paper, we develop an automated technique for *stepwise* model selection (SMS) over time, thereby tackling the challenge of optimal models evolving through-

Proceedings of the 23rd International Conference on Artificial Intelligence and Statistics (AISTATS) 2020, Palermo, Italy. PMLR: Volume 108.

out a dataset. We propose a novel BO algorithm for SMS, treating the prediction performance at each time step as its own black-box function. To solve the resulting multiple black-box function optimization problem *jointly* and *efficiently*, we exploit correlations among black-boxes via deep kernel learning (DKL). Using real-world datasets in healthcare, we verify that our method outperforms both standard BO and multi-objective BO algorithms on a variety of sequence prediction tasks. To the best of our knowledge, we are the first to formulate the problem of stepwise model selection for sequence prediction, and to design and demonstrate an efficient algorithm for this purpose. Our technique contributes to AutoML in developing powerful sequence models while keeping the human out of the loop.

2 Problem Formulation

To establish notation, we first introduce the underlying sequence prediction task, and formalize the stepwise model selection problem that we address in this paper.

Sequence Prediction. Let $\mathbf{o}_t \in \mathbb{R}^d$ denote (observed) input variables, and $e_t \in \mathbb{R}$ the (emitted) output variable, where $t \in \{1, ..., T\}$ in sequences of up to length T. At every time t, the underlying task is to predict the label e_t on the basis of the sequence of observations available up until time t: $(\mathbf{o}_\tau)_{\tau=1}^t$. To this end, we are given a finite dataset $\mathcal{D} = \{(\mathbf{o}_{i,t}, e_{i,t})_{t=1}^T\}_{i=1}^I$ for training purposes, where individual sequences are indexed by $i \in \{1, ..., I\}$ in a dataset with I sequences. Let $\mathbb{X}$ denote the space of *hyperparameters* for such sequence prediction models, including discrete and continuous variables that configure architectures and training. For example, the hyperparameter space of a recurrent neural network (RNN) may include—among others—the size of the hidden state, dropout rate, and coefficient on weight-decay. (If we were to consider different classes of models entirely, e.g. GRUs vs. LSTMs, this can also be accommodated via additional categorical dimensions).

Stepwise Model Selection. In standard Bayesian optimization, the task is to minimize (or maximize) some black-box function $f : \mathbb{X} \to \mathbb{R}$. Let $a_f : \mathbb{X} \to \mathbb{R}$ denote the acquisition function, which captures the utility of evaluating f at $\mathbf{x} \in \mathbb{X}$. At each BO iteration, we use a_f to determine the next point to evaluate, and the goal is to find the global minimizer (or maximizer) of f after the fewest iterations. Concretely, let $\mathcal{D}_{\leq t} = \{(\mathbf{o}_{i,\tau}, e_{i,\tau})_{\tau=1}^t\}_{i=1}^I$ give the filtration of the full dataset $\mathcal{D}$ with respect to time t, and let $\mathcal{L}_t$ denote the validation performance metric of interest (e.g. likelihood of the data, area under the receiver operating characteristic, etc.) for time step t. The (conventional) *single-model* approach for sequence prediction is to find a single maximizer $\mathbf{x}^*$ that is used for *all* time steps t,

$$\mathbf{x}^* \in \arg\max_{\mathbf{x} \in \mathbb{X}} \sum_{t=1}^{T} \mathcal{L}_t(\mathbf{x}, \mathcal{D}^{\text{train}}, \mathcal{D}^{\text{valid}}_{\leq t}) \tag{1}$$

where superscripts on $\mathcal{D}$ denote training and validation splits. Defining $f(\mathbf{x}) = \sum_{t=1}^T \mathcal{L}_t(\mathbf{x}, \mathcal{D}^{\text{train}}, \mathcal{D}^{\text{valid}}_{\leq t})$ gives us the black-box function to be optimized using BO.

In this paper, we extend this formulation to accommodate the SMS problem—that is, of selecting the best sequence prediction model for *each* time step. To this end, we treat the prediction performance at each step $t \in \{1, ..., T\}$ as its own a black-box function f_t. Our objective is to find the best $\mathbf{x}_t^*$ that maximizes each f_t; in other words, we want the set of *stepwise* maximizers,

$$\{\mathbf{x}_t^*\}_{t=1}^T \in \arg\max_{\{\mathbf{x}_t\}_{t=1}^T \in \mathbb{X}^T} \sum_{t=1}^{T} \mathcal{L}_t(\mathbf{x}_t, \mathcal{D}^{\text{train}}, \mathcal{D}^{\text{valid}}_{\leq t}) \tag{2}$$

where for brevity we use $\mathbb{X}^T$ to denote $\prod_{t=1}^T \mathbb{X}$. Defining $f_t(\mathbf{x}_t) = \mathcal{L}_t(\mathbf{x}_t, \mathcal{D}^{\text{train}}, \mathcal{D}^{\text{valid}}_{\leq t})$ for $t \in \{1, ..., T\}$ then gives T black-box functions to be optimized using BO.

Multiple Black-Boxes. Two points require emphasis. The first concerns the problem (SMS), and the second motivates our solution (DKL). First, the T black-box functions are in general *distinct*. In the presence of potential distribution shifts, it is highly unlikely that the optimizer for all t will be the same exact model. For instance, the optimal RNN for the first 24 hours of an ICU physiological stream may require little recurrent memory as the typical patient is very stable; yet as more patients enter deteriorating states over time, we may require more complex hidden states that can better capture both short and long-range patterns. Reducing the SMS problem in (1) to a single black-box problem as in (2) may be overly constraining; we will observe examples of this in our experiments later in Section 5.

Second, however, the T black-box functions are in general *not independent*. For one, we expect a given model's performance to be correlated across time—especially between neighboring steps. Furthermore, instead of obtaining a single point $(\mathbf{x}_n, y_n = f(\mathbf{x}_n))$ per acquisition step n, here we obtain a total of T points $\{(\mathbf{x}_{n,t}, y_{n,t})\}_{t=1}^T$ per acquisition; these are obtained *simultaneously*, since we can observe a model's performance for all times t for every evaluation (i.e. with a single pass through $\mathcal{D}$). Denote by $\mathcal{A}_t = (\mathbf{X}_t, \mathbf{y}_t)$ the acquisition set for time t, where the n-th row of $\mathbf{X}_t$ corresponds to $\mathbf{x}_{n,t}$, and the n-th entry of $\mathbf{y}_t$ to $y_{n,t}$. The number of rows in $\mathbf{X}_t$ equals N, the total number of acquisitions—the same for all functions f_t. In our proposed solution, we will leverage the correlations between time steps t to jointly optimize all $f_1, ..., f_T$, as well as acquiring models via a soft policy prioritizing black-boxes with the highest expected improvement.

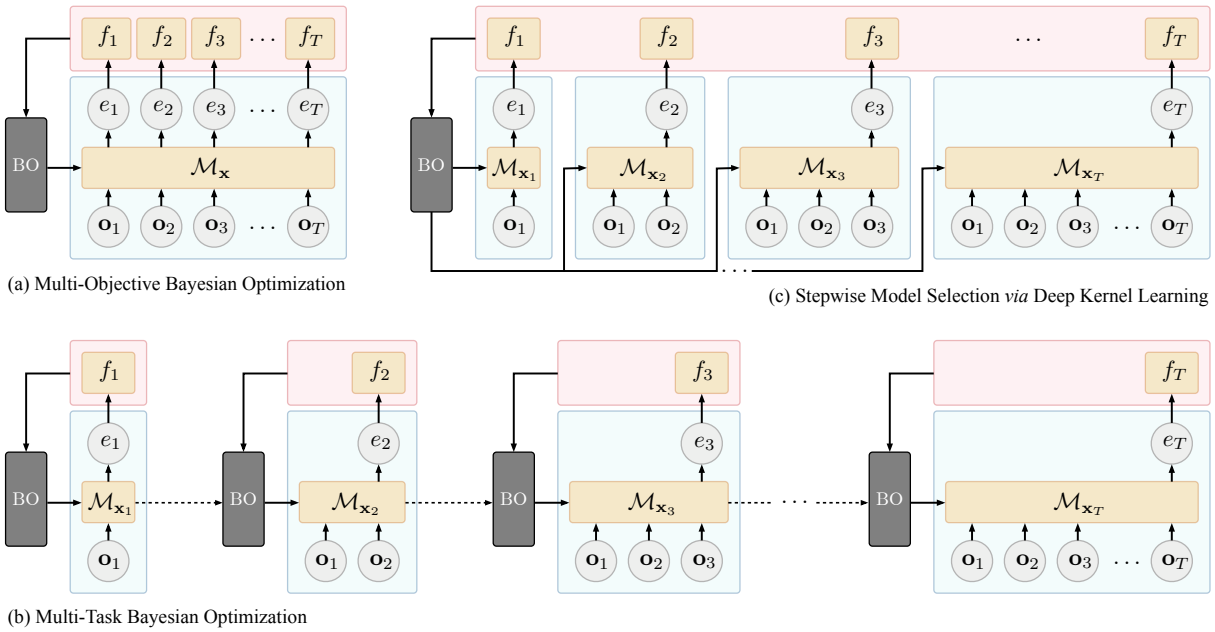


Figure 1: Comparison of related methods in the context of model selection for sequence prediction. Each $\mathcal{M}_{\mathbf{x}}$ indicates a model (hyper-)parameterized by $\mathbf{x}$. (a) Multi-objective Bayesian optimization, which is constrained to learn a single model for all time steps. (b) Multi-task Bayesian optimization, which can be applied *sequentially* across time steps. (c) Our proposed technique for stepwise model selection via deep kernel learning, which *jointly* learns all models for all time steps.

3 Related Work

We take on the problem of selecting sequence prediction models for each time step, casting this as a multiple black-box optimization problem. As such, our work bears some resemblance to multi-objective Bayesian optimization, and to multi-task Bayesian optimization.

Multi-Objective Bayesian optimization (MOBO) Emmerich and Klinkenberg (2008); Hernández-Lobato et al. (2016); Knowles (2006); Picheny (2015); Ponweiser et al. (2008); Shah and Ghahramani (2016); Zitzler and Thiele (1999) deals with optimizing multiple objectives in a trade-off relationship. Consider two objectives

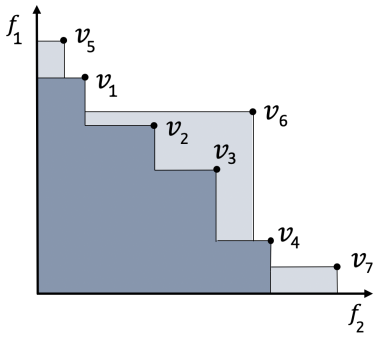


Figure 2: Example of Pareto frontier estimates with two objectives. In the MOBO setting we would often find v_6 the most attractive *single* solution, while in the SMS setting we would be *simultaneously* interested in both v_5 and v_7.

f_1, f_2 as in Figure 2. Suppose that, after a certain number of BO iterations, our current best estimate of the Pareto frontier is given by points v_1 through v_4 (the feasible region is shaded in dark). Further suppose that, as more points are sampled, the frontier is pushed outward by additional points v_5 through v_7. In the MOBO setting, v_6 would often provide the most attractive trade-off between the two objectives—for instance, based on hyper-volume gain. In contrast, our goal in the SMS setting is to find an optimal model $\mathbf{x}_t^*$ at *each* time t; importantly, there is no trade-off relationship—each such $\mathbf{x}_t^*$ does *not* need to be optimal for any other time step. In this example, our primary interests are therefore in v_5 (for f_1) and v_7 (for f_2), but not v_6. If (for whatever reason) we were constrained to select a single prediction model for all time steps, then the SMS problem would be reduced to MOBO.

Multi-Task Bayesian Optimization (MTBO) Perrone et al. (2018); Swersky et al. (2013); Zhang et al. (2017, 2019) deals with transferring knowledge gained from previous optimizations to new tasks, such that subsequent optimizations are more efficient. This setting applies, for example, to the problem where successive batches of (static) data are available or accumulated over time, such that prior trained models may require retraining. Of course, the SMS problem can be (naively) reduced to an MTBO problem—that is, we can optimize all models f_t *sequentially*, for instance by

Table 1: Comparison of related methods in the context of model selection for sequence prediction. [1]Note that correlations *within* black-box functions are exploited in all BO methods; SMS-DKL additionally exploits correlations *across* functions. [2]Some MOBO methods achieve this, but they are *not* scalable to problems with large numbers of objectives (see Section 5).

	Optimization Problem	Number of Optimizers	Optimize All Functions Jointly	Exploit Correlations among All Functions[1,2]	Prioritize which Functions to Optimize
MOBO	$\mathbf{x}^* \in \arg\max_{\mathbf{x}\in\mathbb{X}} \sum_{t=1}^{T} f_t(\mathbf{x})$	1	✓	✗	✗
MTBO	for $t \in \{1,...,T\}$: $\mathbf{x}_t^* \in \arg\max_{\mathbf{x}_t\in\mathbb{X}} f_t(\mathbf{x}_t)$	T	✗	✓	✗
SMS-DKL	$\{\mathbf{x}_t^*\}_{t=1}^T \in \arg\max_{\{\mathbf{x}_t\}_{t=1}^T\in\mathbb{X}^T} \sum_{t=1}^{T} f_t(\mathbf{x}_t)$	T	✓	✓	✓

using optimizations of black-box functions $f_1, ..., f_{t-1}$ to warm-start the optimization for f_t. However, this approach is of little practical interest. Evaluating deep learning models on large datasets is expensive, and in practice we have a limited computational budget—we are interested in finding a good model after a set number of BO evaluations (depending on the dimension of the hyperparameter space). Conducting SMS by reduction to MTBO requires T separate BO procedures in a sequence, and it is unclear how to allocate evaluations among these subproblems while keeping the human out of the loop. In addition, in contrast to the *joint* approach of our proposed solution (which involves a single BO procedure for all f_t), MTBO does not take full advantage of information from all acquisition functions.

In this paper, we take on the SMS problem for sequence prediction models by optimizing all functions $f_1, ..., f_T$ *at the same time*. In contrast to MOBO, we are not constrained by trade-offs between competing objectives. And in contrast to MTBO, our goal is to take full advantage of the potential correlations among black-box functons f_t, as well as information from the acquisition functions, by learning the models for all steps jointly. See Figure 1 and Table 1 for a comparison of MBTO, MOBO, as well as our proposed approach—SMS-DKL[1].

4 SMS via Deep Kernel Learning

We now develop our proposed technique for the SMS problem: a novel BO algorithm that uses deep kernel learning (DKL) to solve the multiple black-box optimization problem jointly and efficiently. Recall that each black-box function f_t corresponds to the validation performance $\mathcal{L}_t$. This depends on the filtration $\mathcal{D}_{\leq t}$ and the selected model $\mathbf{x}_t$; accordingly, we expect the similarity among black-box functions $f_1, ..., f_T$ to be explained by interactions between filtrations and models. Kernels are often deployed as measures of similarity: They are used in support vector machines and gaussian processes to measure the similarity $K(\mathbf{v}, \mathbf{v}')$ between two vectors (using an inner product in a transformed space); they can also be used to measure the similarity $K(p, p')$ between two distributions (using the sample average of an inner kernel, for instance) Muandet et al. (2012). Here, we propose a deep kernel learning method designed to measure the similarity between two *filtration-and-model* tuples. Let $(\mathcal{D}_{\leq t}, \mathbf{x}_t)$ and $(\mathcal{D}_{\leq t'}, \mathbf{x}'_{t'})$ be two such pairs; we will allow their similarity to be captured by a learned kernel parameterized by a neural network.

Section 4.1 describes how to capture similarities between black-boxes, Section 4.2 covers learning and inference, and Section 4.3 provides the acquisition method.

4.1 Deep Kernels

Vector Embedding. We start by transforming tuples $(\mathcal{D}_{\leq t}, \mathbf{x}_t)$ into *fixed-length* vector representations $\mathbf{g}_t$; these are the feature maps that will subsequently be used by the kernel to measure similarities between such tuples. This is accomplished through a neural network that consists of three components. First, an RNN learns a per-instance representation of $\mathcal{D}_{\leq t}$—that is, of $(\mathbf{o}_{i,\tau}, e_{i,\tau})_{\tau=1}^t$ for all $i \in \{1, ..., I\}$. Denote by $\mathbf{h}_{i,t}$ the embedding for instance i; the result of this step is therefore given by the matrix $\mathbf{H}_t$. Second, we pass $\mathbf{H}_t$ through a DeepSets network Zaheer et al. (2017) in order to obtain a *permutation-invariant* embedding, which we denote by $\mathbf{z}_t$. This is important: we want an embedding that is independent of how the individual rows are ordered in $\mathbf{H}_t$. Third, the vectors $\mathbf{z}_t, \mathbf{x}_t$ are concatenated and fed into a multilayer perceptron (MLP) that generates the final feature map $\mathbf{g}_t$. See Figure 3 for a block-diagram.

In DKL, $\mathbf{z}_t$ can be interpreted by analogy to a set of informative statistics on $\mathcal{D}_{\leq t}$; then the MLP component simply operates as a standard deep kernel machine Al-Shedivat et al. (2016); Wilson et al. (2016) that takes vectors $\mathbf{z}_t \oplus \mathbf{x}_t$ as input. Importantly, instead of relying

[1]An implementation of SMS-DKL is available at https://bitbucket.org/mvdschaar/mlforhealthlabpub.

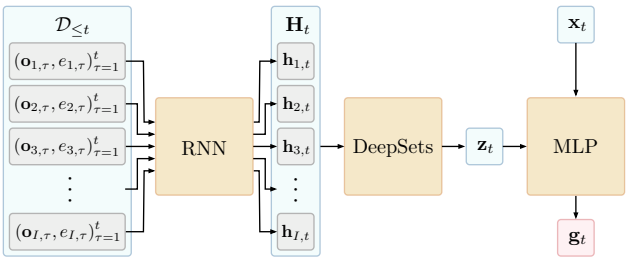


Figure 3: Architecture of DKL network.

on predefined measures of the data or handcrafted statistical meta-features to explain model performance Bardenet et al. (2013); Feurer et al. (2015); Pfahringer et al. (2000), here we allow informative measures to be flexibly learned by way of neural networks.

Kernel Construction. We now show how to use the feature map $\mathbf{g}_t$ to construct a deep kernel. Recall that the acquisition set for time t is given by $\mathcal{A}_t = (\mathbf{X}_t, \mathbf{y}_t)$; here $\mathbf{X}_t$ contains N rows (and $\mathbf{y}_t$ contains N entries), where N is the number of acquisitions made so far. Let $\mathbf{G}_t$ denote the $N \times D$ matrix where the n-th row is given by (D-dimensional) $\mathbf{g}_{n,t}$, the feature map corresponding to $\mathbf{x}_{n,t}$. The deep linear kernel machine is constructed by performing a Bayesian linear regression on $\mathbf{g}_t$—that is, by marginalizing out the output layer $\mathbf{w}_t$ on the top of the feature map $\mathbf{g}_t$, with respect to the posterior distribution of $\mathbf{w}_t$. The likelihood function is as follows,

$$P(\mathbf{y}_t|\mathbf{X}_t, \mathbf{w}_t, \mathbf{\Theta}_t) = \prod_{n=1}^{N} \mathcal{N}(y_{n,t}; \mathbf{G}_t\mathbf{w}_t, \beta_t^{-1}) \tag{3}$$

where β_t is the precision parameter. The prior distribution is $P(\mathbf{w}_t|\lambda_t) = \mathcal{N}(\mathbf{0}, \lambda_t^{-1}\mathbf{I}_{D\times D})$ with precision parameter λ_t, which leads to the posterior distribution,

$$P(\mathbf{w}_t|\mathcal{A}_t, \mathbf{\Theta}_t) = \mathcal{N}(\mathbf{m}_{\mathbf{w}_t}, \mathbf{K}_{\mathbf{w}_t}^{-1}) \tag{4}$$

where the set $\mathbf{\Theta}_t$ collects all neural network parameters in the DKL architecture as well as λ_t and β_t, and the mean function $\mathbf{m}_{\mathbf{w}_t}$ and kernel $\mathbf{K}_{\mathbf{w}_t}$ are as follows,

$$\mathbf{m}_{\mathbf{w}_t} = \frac{\beta_t}{\lambda_t}\mathbf{K}_{\mathbf{w}_t}^{-1}\mathbf{G}_t^\top\mathbf{y}_t, \quad \mathbf{K}_{\mathbf{w}_t} = \frac{\beta_t}{\lambda_t}\mathbf{G}_t^\top\mathbf{G}_t + \mathbf{I}_{D\times D}$$

4.2 Learning and Inference

We learn the parameters in $\mathbf{\Theta}_t$ by marginal likelihood optimization. We switch between the primal and dual forms of the log marginal likelihood for computational efficiency and numerical stability. In its primal form, the log marginal likelihood is given by the following,

$$\begin{aligned}\mathcal{L}(\mathbf{\Theta}_t) = &-\frac{N}{2}\log(2\pi\beta_t^{-1}) - \frac{\beta_t}{2}\|\mathbf{y}_t\|^2 \\ &+ \frac{\beta_t^2}{\lambda_t}\mathbf{y}_t^\top\mathbf{G}_t\mathbf{K}_{\mathbf{w}_t}^{-1}\mathbf{G}_t^\top\mathbf{y}_t - \frac{1}{2}\log|\mathbf{K}_{\mathbf{w}_t}|\end{aligned}$$

In its dual form, the log marginal likelihood is given as the logarithm of $\mathcal{N}(\mathbf{y}_t; \lambda_t^{-1}\mathbf{G}_t\mathbf{G}_t^\top + \beta_t^{-1}\mathbf{I}_{N\times N})$. When $N > D$, we optimize the log marginal likelihood in primal form, otherwise in dual form. Now, our operating assumption is that the black-box functions $f_1, ..., f_T$ are *correlated* in some way; accordingly, we let the neural network parameters be *shared* over all the time steps t, giving the multi-task marginal likelihood Perrone et al. (2018),

$$\mathcal{L}(\mathbf{\Theta}) = \sum_{t=1}^{T}\mathcal{L}(\mathbf{\Theta}_t) \tag{5}$$

where we have used $\mathbf{\Theta}$ to indicate $\cup_{t=1}^{T}\mathbf{\Theta}_t$. The overall computational complexity of optimizing $\mathcal{L}(\mathbf{\Theta})$ is $O(T\max\{N, D\}(\min\{N, D\})^2)$. If T is very large, then we can first randomly sample a subset of time steps $S \subset \{1, ..., T\}$, and then maximize $\sum_{t\in S}\mathcal{L}(\mathbf{\Theta}_t)$ instead at each iteration of marginal likelihood optimization.

Acquisition Function. To construct the acquisition function $a_{f,t}(\mathbf{x}_t^\dagger|\mathcal{A}_t)$ in BO (for test data point $\mathbf{x}_t^\dagger$), we first obtain the feature map $\mathbf{g}_t$ by passing $\mathbf{x}_t^\dagger$ through the neural network. Then the predictive distribution is obtained by integrating out $\mathbf{w}_t$ in the delta measure $\delta(f_t(\mathbf{x}_t^\dagger) = \mathbf{w}_t^\top\mathbf{g}_t)$ with respect to its posterior in (4):

$$f_t(\mathbf{x}_t^\dagger) \sim \mathcal{N}\big(\mu(\mathbf{x}_t^\dagger|\mathcal{A}_t, \mathbf{\Theta}_t), \sigma^2(\mathbf{x}_t^\dagger|\mathcal{A}_t, \mathbf{\Theta}_t)\big) \tag{6}$$

where

$$\begin{aligned}\mu(\mathbf{x}_t^\dagger|\mathcal{A}_t, \mathbf{\Theta}_t) &= \mathbf{m}_{\mathbf{w}_t}^\top\mathbf{g}_t \\ \sigma^2(\mathbf{x}_t^\dagger|\mathcal{A}_t, \mathbf{\Theta}_t) &= \frac{1}{\lambda_t}\mathbf{g}_t^\top\mathbf{K}_{\mathbf{w}_t}^{-1}\mathbf{g}_t\end{aligned} \tag{7}$$

4.3 Acquisition Method

In standard BO, the next model to be acquired is chosen by maximizing the acquisition function $a_f(\mathbf{x}|\mathcal{A})$—e.g. the probability of improvement (PI) Kushner (1964), expected improvement Jones et al. (1998); Mockus et al. (1978), Gaussian process upper confidence bound (GP-UCB) Srinivas et al. (2009), and entropy search (ES) Hennig and Schuler (2012):

$$\mathbf{x}^* \in \arg\max_{\mathbf{x}\in\mathbb{X}} a_f(\mathbf{x}|\mathcal{A}) \tag{8}$$

In DKL, the first two components produce the embedding vector $\mathbf{z}_t$. The third component is what takes $\mathbf{x}_t$ as input (along with $\mathbf{z}_t$), and its output is what the posterior in (6) and corresponding acquisition function are constructed with; we can optimize $a_{f,t}(\mathbf{x}_t, \mathcal{A}_t)$ by optimizing the input $\mathbf{x}_t$ only in the third network.

In our problem, we actually need to optimize T black-box functions $f_1, ..., f_T$ at the same time. In addition, recall that we can observe a given model's performance for *all* time steps t after every single model evaluation (i.e. we obtain T data points $\{(\mathbf{x}_{n,t}, y_{n,t})\}_{t=1}^{T}$ per acquisition). One straightforward solution is to simply define

the sum $f_{\text{sum}} = \sum_{t=1}^{T} f_t$ and make acquisitions on the basis of this sum. However, this is not desirable since the optimizer of f_{sum} is in general not identical to the optimizer of *each* individual f_t. Optimizing f_{sum} is only suitable if we were constrained to select a single model for all time steps t (which we are not). In our case, at each BO iteration we first compute the *individual* optimizers $\mathbf{x}_t^*$ for each acquisition function $a_{f,t}(\mathbf{x}_t|\mathcal{A}_t)$, $t \in \{1, ..., T\}$. Then, our choice c of which specific $\mathbf{x}_t^*$ to acquire is made via the following a stochastic policy,

$$p(c = \mathbf{x}_t^*|\mathbf{a}) = \frac{a_{f,t}(\mathbf{x}_t^*|\mathcal{A}_t)}{\sum_{\tau=1}^{T} a_{f,\tau}(\mathbf{x}_\tau^*|\mathcal{A}_\tau)} \tag{9}$$

where $\mathbf{a} = [a_{f,1}(\mathbf{x}_1^*|\mathcal{A}_1), ..., a_{f,T}(\mathbf{x}_T^*|\mathcal{A}_T)]^\top$. This acquisition method is inspired by the Hedge algorithm Freund and Schapire (1999). We treat each acquisition function as an expert giving advice as to which model to acquire. Assuming that we believe equally in all experts throughout the BO experiment, the probability of following the advice of the t-th expert is given by (9). Algorithm 1 in Appendix E provides pseudo- code summarizing our proposed method (SMS-DKL).

5 Experiments and Discussion

Sequence prediction admits a variety of models, among the most popular being RNNs in machine learning, although the difficulties of training them are widely recognized Pascanu et al. (2013). In this paper, we are motivated by the problem of temporal distribution shift within a dataset, a phenomenon especially relevant in the medical setting Oh et al. (2019), and in the presence of which the optimal model itself may vary over time. So far, we have formalized this challenge as one of stepwise model selection (SMS), and proposed a solution via deep kernel learning (DKL). Three questions remain, and our goal in this section is to answer them:

- First, *why* do we expect to benefit from stepwise selection at all? While the abstract notion of potential distribution shifts gives some intuition, here we empirically illustrate the validity of SMS *as the problem*: We observe improvements simply by applying post-hoc stepwise selection over standard BO and MOBO.
- Second, *what* is the practical benefit our technique for model selection? Here, we demonstrate the consistent, significant advantage of DKL *as the solution*: We observe a clear improvement by addressing stepwise selection directly in the optimization procedure.
- Third, *how* do the correlations ultimately influence the optimal models selected? Here, we visualize the correlations in model performance over time, as well as the learned embeddings $\mathbf{z}_t$ and optimizers $\mathbf{x}_t$, shedding further light on the workings of SMS-DKL.

Datasets. We use three datasets in our experiments. The first consists of patients enrolled in the UK Cystic Fibrosis registry (**UKCF**), which records annual follow-up trajectories for over 10,000 patients in 2008–2015. At each time step, we issue predictions on the basis of 90 temporal variables (e.g. treatments, comorbidities, infections), focusing on three important clinical outcomes (see e.g. Alaa and van der Schaar (2019)): the 1-year mortality (1YM), allergic broncho-pulmonary aspergillosis (ABPA), and the lung infection E. coli. The second consists of patients in intensive care units from the MIMIC-III database (**MIMIC**), containing physiological data streams for over 22,000 patients. During the first 48 hours of each episode, we issue predictions using 40 temporal variables (including the most frequently measured vital signs and lab tests) focusing on three important clinical outcomes (see e.g. Oh et al. (2019)): acute respiratory failure (ARF), shock, and in-hospital mortality (IHM). The third (**WARDS**), assembled by Alaa et al. (2017), consists of over 6,000 patients hospitalized in the general medicine floor of a major medical center in 2013–2015. On the basis of 21 physiological data streams (including vital signs and lab tests), we predict whether each patient will be admitted to critical care within 24 hours from the current time as a result of clinical deterioration (ICU).

Experimental Setup. We have a total of 7 sequence prediction tasks from the three datasets. Sequences are of length 6 (at 1-year resolution) for UKCF, length 24 (at 2-hour resolution) for MIMIC, and length 24 (at 1-hour resolution) for WARDS. Benchmarks are implemented using the `GPyOpt` library or original source code. In SMS-DKL, the RNN component is implemented using LSTMs, the DeepSets component as a ReLU network (with an output embedding $\mathbf{z}_t$ or $\mathbf{z}_{t,m}$ of size 1), and the MLP component as a feedforward tanh network. See Appendix A for additional details on implementation. The underlying search space is the space of RNN models for sequence prediction, and hyperparameters considered include the learning rate, batch size, training epochs, hidden state size, input dropout rate, recurrent dropout rate, and the ℓ_2-regularization coefficient. See Appendix B for additional details on hyperparameter space. In the presence of label imbalance in the data (common to the medical setting; see Table 4 in Appendix E) , we focus on optimizing the area under the precision-recall curve (AUPRC) as performance metric. BO is carried out to a maximum of 500 iterations. Each experiment is repeated for a total of 10 times, each with a different random training and validation split. Each run of the experiment uses a different random seed to initialize the BO algorithms, and the same random seed is used for all algorithms at each run.

Benchmarks. We compare the proposed SMS-

DKL with a standard BO algorithm and two MOBO algorithms ParEGO Knowles (2006) and PESMO Hernández-Lobato et al. (2016) on the SMS problem. While other MOBO algorithms are available (such as EHI Emmerich and Klinkenberg (2008), SMSEGO Ponweiser et al. (2008), and SUR Picheny (2015)), they are *not* scalable to problems with a large number of objectives. In particular, SMSEGO and EHI make acquisitions by computing the hyper-volume gain, which is expensive in high dimensions (i.e. large numbers of objectives). Similarly, SUR is an extremely expensive criterion only feasible for 2 or 3 objectives at most, because it computes the expected decrease in the area under the probability of improving the hyper-volume.

• **GP**. We first consider a standard BO algorithm (GP) for comparison; this operates by simply optimizing the sum of the model performance metric over all steps t.

• **ParEGO** first transforms the multi-objective problem into a single-objective problem: At each BO iteration, the multiple objectives f_t are scalarized into $f_{\boldsymbol{\theta}}$ using a randomly sampled weight vector $\boldsymbol{\theta} = (\theta_1, ..., \theta_T)$,

$$f_{\boldsymbol{\theta}}(x) = \max_{t \in T}(\theta_t f_t(x)) + 0.05 \sum_{t=1}^{T} \theta_t f_t(x) \qquad (10)$$

Then at each BO iteration, a standard acquisition function can be used on $f_{\boldsymbol{\theta}}(x)$ to select the next point.

• **PESMO** is a recent, state-of-the-art MOBO algorithm based on predictive entropy search. The acquisition function in PESMO is expressed by the following,

$$a(x) = H(\mathcal{X}^*|\mathcal{D}) - \mathbb{E}_y[H(\mathcal{X}^*|\mathcal{D} \cup \{(\mathbf{x}, y)\})] \qquad (11)$$

where $H(\cdot)$ denotes the entropy and $\mathcal{X}^*$ is the Pareto set. PESMO operates by selecting the point that maximizes the information gain with respect to the Pareto set.

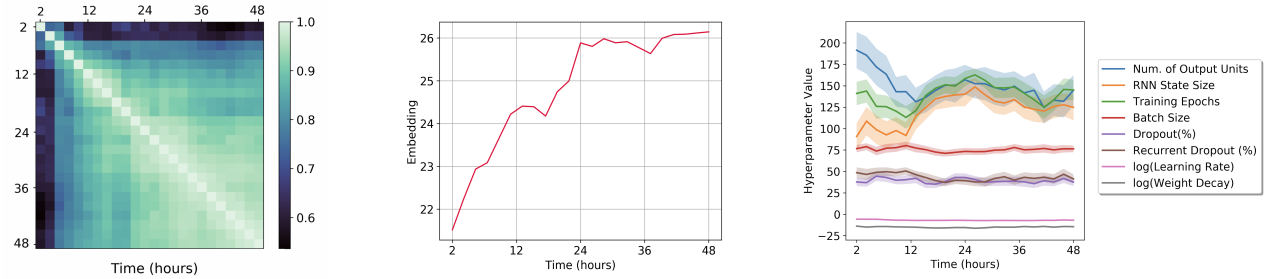
• **Post-hoc Stepwise Variants**. Since the goal in SMS is to attain good performance for *each* individual time step, we additionally consider a straightforward modification to all aforementioned benchmarks that applies an extra post-hoc selection step, choosing (among all models) the best-performing model on a *per-step* basis. For each of the algorithms considered (GP, ParEO, and PESMO), we denote by subscript "WISE" the results for this "stepwise" modification to the benchmark.

5.1 Experiment Results

Overall Results. For all BO algorithms considered, Table 2 reports the AUPRC score (averaged over all time steps t) at the 100-th and 500-th BO iteration. We now answer the first question posed at the beginning of this section: Is it reasonable to expect to benefit from SMS (at all)? Comparing each benchmark with its post-hoc stepwise modification, we answer in the affirmative: Across all benchmarks, observe that prediction performance is invariably improved simply by going back and selecting the best model on a per-step basis. The second—and perhaps more interesting—question is whether solving the SMS problem directly within the optimization procedure can offer additional gains. Comparing our proposed DKL solution with any comparator, the answer is also in the affirmative: Observe that SMS-DKL consistently and significantly outperforms both standard BO and MOBO algorithms across all datasets and prediction targets, at both the 100-th and 500-th BO iteration mark. This is true regardless of whether we allow comparators the additional

Dataset	UKCF			MIMIC			WARDS
Target	1YM	ABPA	E. coli	IHM	Shock	ARF	ICU
100 BO Iterations							
GP	.586 ± .001	.627 ± .001	.872 ± .001	.460 ± .003	.107 ± .001	.114 ± .001	.161 ± .004
GP_{WISE}	.593 ± .002	.636 ± .001	.877 ± .001	.463 ± .002	.113 ± .001	.126 ± .001	.168 ± .005
ParEGO	.588 ± .002	.623 ± .002	.872 ± .002	.461 ± .001	.107 ± .000	.118 ± .001	.176 ± .003
$\text{ParEGO}_{\text{WISE}}$	.594 ± .001	.633 ± .003	.876 ± .001	.464 ± .001	.112 ± .001	.124 ± .001	.184 ± .004
PESMO	.592 ± .002	.629 ± .001	.874 ± .001	.467 ± .001	.111 ± .002	.111 ± .002	.176 ± .006
$\text{PESMO}_{\text{WISE}}$	.598 ± .002	.639 ± .001	.878 ± .001	.469 ± .001	.115 ± .001	.123 ± .001	.182 ± .005
SMS-DKL	**.601 ± .001**	**.641 ± .001**	**.880 ± .001**	**.474 ± .002**	**.118 ± .001**	**.128 ± .001**	**.197 ± .004**
500 BO Iterations							
GP	.592 ± .001	.633 ± .002	.876 ± .000	.471 ± .001	.113 ± .001	.120 ± .001	.176 ± .003
GP_{WISE}	.601 ± .001	.644 ± .001	.882 ± .000	.475 ± .001	.122 ± .001	.135 ± .001	.189 ± .002
ParEGO	.593 ± .002	.632 ± .001	.875 ± .001	.469 ± .001	.114 ± .001	.122 ± .002	.187 ± .004
$\text{ParEGO}_{\text{WISE}}$	.601 ± .001	.644 ± .001	.881 ± .001	.473 ± .001	.121 ± .001	.136 ± .001	.199 ± .002
PESMO	.592 ± .001	.632 ± .002	.876 ± .000	.469 ± .001	.113 ± .002	.117 ± .002	.186 ± .004
$\text{PESMO}_{\text{WISE}}$	.601 ± .001	.642 ± .001	.881 ± .000	.474 ± .001	.119 ± .001	.132 ± .001	.194 ± .002
SMS-DKL	**.603 ± .001**	**.645 ± .001**	**.884 ± .000**	**.476 ± .001**	**.123 ± .001**	**.138 ± .001**	**.207 ± .002**

Table 2: Performance of SMS-DKL and Benchmarks: AUPRC scores at the 100-th and 500-th BO iteration mark.



(a) Correlation in black-boxes f_t (b) Change in dataset embedding $\mathbf{z}_t$ (c) Change in optimal prediction model $\mathbf{x}_t$

Figure 4: Learning correlations and changes over time. SMS-DKL leverages correlations to select stepwise-optimal models.

freedom of choosing the best stepwise models post-hoc.

Correlations and Changes. Finally, how do changes and correlations over time ultimately play out with respect to optimal models? In SMS-DKL, the source of efficiency in optimization stems from its ability to learn the similarities and differences between black-box functions f_t. Using the prediction of shock in MIMIC as an example, Figure 4a shows a correlation matrix of model performance across the 48-hour interval considered, at the 500-th iteration mark. Here we see that black-box functions within first 12 hours are weakly correlated with subsequent times, and those within the latter half of the interval appear strongly correlated with each other—with values exceeding 0.95. Observe that this pattern is automatically picked up and reflected in the learned embeddings $\mathbf{z}_t$ of the datasets $\mathcal{D}_{\leq t}$ themselves: Figure 4b shows these (one-dimensional) embeddings $\mathbf{z}_t$ over time. Notice a clear evolution consistent with the previous observation: the evolving datasets are autocorrelated, with particularly strong similarities among the latter 24-hour interval. Importantly, this plays out with respect to optimizers $\mathbf{x}_t$ in parallel: Figure 4c shows the evolution of optimal models $\mathbf{x}_t$ over time; in order to highlight the trend across time steps, values are averaged over the 20 highest-performing models. (The conventional approach of selecting a single model would correspond to a series of flat lines). Consistent with our intuitions, notice—for instance—that models for earlier steps (which have access to less temporal information) appear to require less recurrent memory.

As an additional sanity check, Table 4 in Appendix E also shows the first-order autocorrelations of input features in each dataset prediction task; these are first computed per feature, then averaged over all features. We see that the autocorrelations are stronger in MIMIC and WARDS than in UKCF. Although this is (at best) a rough proxy for the performance correlation between models across time steps, we observe an intuitive pattern: SMS-DKL shows more significant gains over datasets with higher autocorrelations. In particular, we outperform all of the benchmarks by the widest margin in WARDS.

5.2 Discussion

The advantage of SMS-DKL for sequence prediction is predicated on the fact that the optimal model for predicting e_t may change with t—within a given dataset. While there may be a variety of reasons for this phenomenon (encapsulated by the general notion of temporal distribution shift Oh et al. (2019)), the benefit here is that optimal models are *automatically* selected over time—agnostic as to the precise underlying mechanism of change, and *without* requiring domain-specific engineering to explicitly model time-varying relationships.

Generalization Performance. Of course, a variety of sophisticated *single-model* techniques can be—and have been—used to tackle temporal distribution shift; these include explicitly including temporal encodings, modeling abrupt transitions, mixing weights over time, as well as learning hypernetworks to modify the weights of the primary RNN model Ha et al. (2016); Oh et al. (2019). On the one hand, such techniques have been shown to outperform the baseline RNN model on held-out test data; see Oh et al. (2019) for a comprehensive analysis. On the other hand, extremely competitive (test set) performance can also be achieved via the simple application of SMS-DKL in optimizing the (unmodified) baseline RNN alone: In Appendix C, we show a head-to-head comparison of generalization performance for all such methods on MIMIC prediction tasks, and observe—interestingly—that SMS-DKL gives either the best or second-best test-set performance—purely by optimizing the baseline RNN model.

In this paper, we formalized the SMS problem in the context of sequence prediction, and developed the DKL algorithm as a solution. Using real-world datasets in healthcare, we illustrated the advantage of SMS-DKL over standard and multi-objective BO approaches for model selection. In contrast to alternative single-model techniques, we further verified the effectiveness of SMS-DKL with respect to generalization—with the added advantage that the method is simple and automatic.

Acknowledgements

This work was supported by GlaxoSmithKline (GSK), Alzheimer's Research UK (ARUK), the US Office of Naval Research (ONR), and the National Science Foundation (NSF): grant numbers ECCS1462245, ECCS1533983, and ECCS1407712. We thank the reviewers for their helpful comments. We thank the UK Cystic Fibrosis Trust, the MIT Lab for Computational Physiology, and Ahmed M. Alaa respectively for making and/or preparing the UKCF, MIMIC, and WARDS datasets available for research.

References

Jasper Snoek, Hugo Larochelle, and Ryan P Adams. Practical bayesian optimization of machine learning algorithms. In *Advances in neural information processing systems*, pages 2951–2959, 2012.

Daniel Hernández-Lobato, Jose Hernandez-Lobato, Amar Shah, and Ryan Adams. Predictive entropy search for multi-objective bayesian optimization. In *International Conference on Machine Learning*, pages 1492–1501, 2016.

Victor Picheny. Multiobjective optimization using gaussian process emulators via stepwise uncertainty reduction. *Statistics and Computing*, 25(6):1265–1280, 2015.

Amar Shah and Zoubin Ghahramani. Pareto frontier learning with expensive correlated objectives. In *International Conference on Machine Learning*, pages 1919–1927, 2016.

Valerio Perrone, Rodolphe Jenatton, Matthias W Seeger, and Cédric Archambeau. Scalable hyperparameter transfer learning. In *Advances in Neural Information Processing Systems*, pages 6845–6855, 2018.

Kevin Swersky, Jasper Snoek, and Ryan P Adams. Multi-task bayesian optimization. In *Advances in neural information processing systems*, pages 2004–2012, 2013.

Yehong Zhang, Trong Nghia Hoang, Bryan Kian Hsiang Low, and Mohan Kankanhalli. Information-based multi-fidelity bayesian optimization. In *Advances in Neural Information Processing Systems*, 2017.

Jeeheh Oh, Jiaxuan Wang, Shengpu Tang, Michael Sjoding, and Jenna Wiens. Relaxed weight sharing: Effectively modeling time-varying relationships in clinical time-series. *arXiv preprint arXiv:1906.02898*, 2019.

Jenna Wiens, John Guttag, and Eric Horvitz. Patient risk stratification with time-varying parameters: a multitask learning approach. *The Journal of Machine Learning Research*, 17(1):2797–2819, 2016.

Michael Emmerich and Jan-willem Klinkenberg. The computation of the expected improvement in dominated hypervolume of pareto front approximations. *Rapport technique, Leiden University*, 34:7–3, 2008.

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

Wolfgang Ponweiser, Tobias Wagner, Dirk Biermann, and Markus Vincze. Multiobjective optimization on a limited budget of evaluations using model-assisteds-metric selection. In *International Conference on Parallel Problem Solving from Nature*, pages 784–794. Springer, 2008.

Eckart Zitzler and Lothar Thiele. Multiobjective evolutionary algorithms: a comparative case study and the strength pareto approach. *IEEE transactions on Evolutionary Computation*, 3(4):257–271, 1999.

Yao Zhang, James Jordon, Ahmed M Alaa, and Mihaela van der Schaar. Lifelong bayesian optimization. *arXiv preprint arXiv:1905.12280*, 2019.

Krikamol Muandet, Kenji Fukumizu, Francesco Dinuzzo, and Bernhard Schölkopf. Learning from distributions via support measure machines. In *Advances in neural information processing systems*, pages 10–18, 2012.

Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabas Poczos, Ruslan R Salakhutdinov, and Alexander J Smola. Deep sets. In *Advances in neural information processing systems*, pages 3391–3401, 2017.

Maruan Al-Shedivat, Andrew Gordon Wilson, Yunus Saatchi, Zhiting Hu, and Eric P Xing. Learning scalable deep kernels with recurrent structure. *arXiv preprint arXiv:1610.08936*, 2016.

Andrew Gordon Wilson, Zhiting Hu, Ruslan Salakhutdinov, and Eric P Xing. Deep kernel learning. In *Artificial Intelligence and Statistics*, pages 370–378, 2016.

Rémi Bardenet, Mátyás Brendel, Balázs Kégl, and Michele Sebag. Collaborative hyperparameter tuning. In *International conference on machine learning*, pages 199–207, 2013.

Matthias Feurer, Aaron Klein, Katharina Eggensperger, Jost Springenberg, Manuel Blum, and Frank Hutter. Efficient and robust automated machine learning. In *Advances in neural information processing systems*, pages 2962–2970, 2015.

Bernhard Pfahringer, Hilan Bensusan, and Christophe G Giraud-Carrier. Meta-learning by landmarking various learning algorithms. In *ICML*, pages 743–750, 2000.

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

Donald R Jones, Matthias Schonlau, and William J Welch. Efficient global optimization of expensive black-box functions. *Journal of Global optimization*, 13(4):455–492, 1998.

Jonas Mockus, Vytautas Tiesis, and Antanas Zilinskas. Toward global optimization, volume 2, chapter bayesian methods for seeking the extremum, 1978.

Niranjan Srinivas, Andreas Krause, Sham M Kakade, and Matthias Seeger. Gaussian process optimization in the bandit setting: No regret and experimental design. *arXiv preprint arXiv:0912.3995*, 2009.

Philipp Hennig and Christian J Schuler. Entropy search for information-efficient global optimization. *Journal of Machine Learning Research*, 13(Jun):1809–1837, 2012.

Yoav Freund and Robert E Schapire. Adaptive game playing using multiplicative weights. *Games and Economic Behavior*, 29(1-2):79–103, 1999.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks. In *International conference on machine learning*, pages 1310–1318, 2013.

Ahmed M. Alaa and Mihaela van der Schaar. Attentive state-space modeling of disease progression. In *2019 Conference on Neural Information Processing Systems*, 2019.

Ahmed M Alaa, Jinsung Yoon, Scott Hu, and Mihaela Van der Schaar. Personalized risk scoring for critical care prognosis using mixtures of gaussian processes. *IEEE Transactions on Biomedical Engineering*, 65 (1):207–218, 2017.

David Ha, Andrew Dai, and Quoc V Le. Hypernetworks. *arXiv preprint arXiv:1609.09106*, 2016.

J González. Gpyopt: A bayesian optimization framework in python, 2016.

Christopher KI Williams and Carl Edward Rasmussen. *Gaussian processes for machine learning*, volume 2. MIT press Cambridge, MA, 2006.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Table 3: Performance of SMS-DKL (optimized over the baseline LSTM model) and Single-model benchmarks (with various modifications to the baseline LSTM model). Results are computed on (held-out) test data; in line with the original paper Oh et al. (2019), we include the area under the receiver operating characteristic (AUROC) and the area under the precision-recall curve (AUPRC), each shown with 95% confidence intervals. Results for SMS-DKL are computed at after 500 BO iterations; note that these numbers (computed on test data) are not exactly identical to those in Table 2 (computed on validation data).

Target	IHM		Shock		ARF	
Metric	AUROC	AUPRC	AUROC	AUPRC	AUROC	AUPRC
LSTM	0.80 [0.78, 0.83]	0.39 [0.33, 0.43]	0.59 [0.49, 0.69]	0.09 [0.05, 0.16]	0.47 [0.35, 0.58]	0.04 [0.02, 0.07]
LSTM+t	0.81 [0.79, 0.83]	0.41 [0.36, 0.47]	0.62 [0.53, 0.70]	0.08 [0.05, 0.15]	0.42 [0.30, 0.54]	0.04 [0.02, 0.07]
LSTM+TE	0.82 [0.80, 0.85]	0.43 [0.38, 0.48]	0.60 [0.50, 0.69]	0.10 [0.06, 0.20]	0.48 [0.35, 0.61]	0.05 [0.03, 0.10]
HyperLSTM	0.82 [0.80, 0.84]	0.42 [0.37, 0.47]	0.63 [0.54, 0.72]	0.08 [0.05, 0.12]	0.57 [0.44, 0.68]	0.06 [0.03, 0.10]
shiftLSTM	0.81 [0.79, 0.84]	0.43 [0.37, 0.48]	0.61 [0.52, 0.70]	0.09 [0.05, 0.16]	0.61 [0.49, 0.70]	0.10 [0.03, 0.21]
mixLSTM	0.83 [0.81, 0.85]	0.45 [0.40, 0.50]	**0.67 [0.58, 0.76]**	0.10 [0.06, 0.16]	**0.72 [0.62, 0.80]**	**0.15 [0.06, 0.27]**
SMS-DKL	**0.84 [0.82, 0.87]**	**0.46 [0.40, 0.53]**	0.65 [0.59, 0.72]	**0.12 [0.07, 0.17]**	0.66 [0.61, 0.72]	0.11 [0.07, 0.15]

A Implementation Details

Implementation. The benchmarks GP and ParEGO (and their stepwise variants) are implemented using the `GPyOpt` library González (2016). All models use a Matérn-5/2 covariance kernel and automatic relevance determination hyperparameters, optimized by empirical Bayes Williams and Rasmussen (2006). For these algorithms as well as SMS-DKL, we use the expected improvement (EI) Jones et al. (1998); Mockus et al. (1978) as the acquisition function. The DKL network consists of three components as shown in Figure 3. The RNN component is implemented as an LSTM network with 50 hidden units per cell, using tanh as hidden recurrent activation and sigmoid as output activation. The DeepSets component is implemented as a four-layer feedforward network with 32 hidden units per layer and ReLU as activation; we average all the samples in the dataset after the transformation of the second hidden layer. See Zaheer et al. (2017) for additional details on DeepSets and permutation invariance. In our experiments, we set the learned embedding of the filtration at each time step to be of size 1. As for the MLP component in the DKL network, we use a three-layer feedforward network with 32 hidden units per layer and tanh as activation. In training, we set the maximum number of training iterations $M = 500$. The optimizer we use is Adam Kingma and Ba (2014). The benchmark PESMO is implemented using the source code provided by Hernández-Lobato et al. (2016).[2] The size of the Pareto set sample in PESMO is 50, which is used and claimed to be suitable for several hundreds of acquisitions in the original PESMO paper. For all BO algorithms, we set the maximum number of function evaluations $N = 500$ and use the same initial sample for all the algorithms.

[2] https://github.com/HIPS/Spearmint/tree/PESM

B Hyperparameter Space

In Section 5, we evaluate the proposed SMS-DKL algorithm and all benchmark algorithms in optimizing the hyperparameters of RNN models on 7 sequence prediction tasks. The RNN architecture used in the experiment is a standard LSTM network with tanh as hidden recurrent activation and a hidden ReLU output layer for making predictions. The full eight-dimensional hyperparameter space $\mathbb{X}$ of RNN models is as follows,

- Number of output units: int, [10, 200]
- RNN state size: int, [10, 300]
- Training epochs: int, [32, 200]
- Batch size: int, [32, 100]
- Dropout rate: float, [0.1, 0.9]
- Recurrent dropout rate: float, [0.1, 0.9]
- Logarithm of learning rate: int, [-8, -3]
- Logarithm of weight decay: int, [-20, 1]

C Generalization Performance

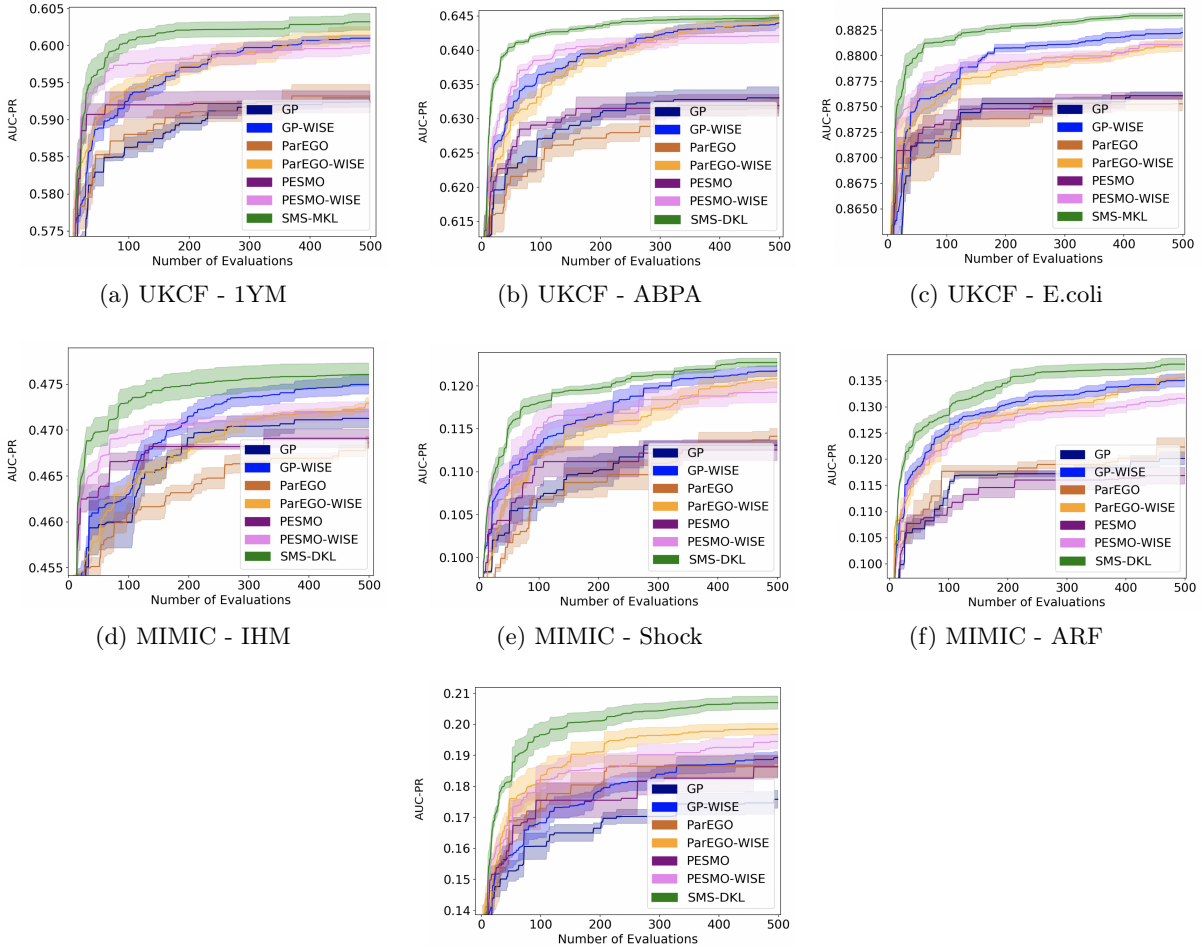
In the main manuscript, we primarily focused on comparing SMS-DKL to other algorithms in the context of BO—that is, with respect to the performance of selected hyperparameters on validation data. Now in practice, a reasonable question is whether these stepwise hyperparameters—selected by SMS-DKL on the validation set—can generalize well to (held-out) test data. In particular, a variety of sophisticated *single-model* techniques have been proposed to address the problem of temporal distribution shift by modifying the baseline LSTM model. These explicitly include

time as a parameter (LSTM+t), incorporate temporal encodings (LSTM+TE), model abrupt transitions (shiftLSTM), mix weights over time (mixLSTM), as well as learning hypernetworks to modify the weights of the LSTM model (HyperLSTM); we refer to Oh et al. (2019) for a more detailed treatment. Here, we include a head-to-head comparison of such single-model techniques with SMS-DKL—applied over the baseline LSTM alone. Table 3 shows results for all aforementioned single-model techniques designed to accommodate time-varying relationships, as well as the result of SMS-DKL (applied to the baseline LSTM model). Results for the single-model techniques are reprinted from Oh et al. (2019), and include all of the same prediction tasks on the MIMIC dataset. Results for SMS-DKL are obtained via the same training and testing splits; we execute the same data processing procedure using the source code accompanying the original paper.[3] On the one hand, the various modifications to the LSTM model clearly improve performance over the baseline LSTM model. On the other hand, we observe that extremely competitive performance is also achieved by the simple application of SMS-DKL in optimizing the baseline LSTM model: SMS-DKL exhibits either the best or second-best test-set performance—purely by optimizing the (unmodified) LSTM baseline. SMS-DKL lays the foundation for AutoML to offer a general and convenient framework of developing powerful sequence prediction models while keeping the human out of the loop.

D Convergence Plots

In this section, we provide the convergence plots of all BO algorithms over all 500 evaluations, for each of the prediction tasks corresponding to Table 2 in Section 5.

[3] https://gitlab.eecs.umich.edu/mld3



(a) UKCF - 1YM (b) UKCF - ABPA (c) UKCF - E.coli

(d) MIMIC - IHM (e) MIMIC - Shock (f) MIMIC - ARF

(g) WARDS - ICU

Figure 5: Convergence plots of all BO algorithms over all 500 evaluations, for each of the seven prediction tasks in Table 2.

E Table and Pseudo code

Dataset	Target	Autocorr.	% Postive (start)	(end)
UKCF	1YM	0.592	12.2	23.4
	ABPA	0.564	27.6	12.7
	E. coli	0.564	52.4	28.5
MIMIC	IHM	0.849	13.2	13.2
	Shock	0.867	8.9	8.9
	ARF	0.875	7.2	7.2
WARDS	ICU	0.976	1.7	3.2

Table 4: Feature autocorrelations and % positive labels. The slight inter-task variation in feature autocorrelations within a dataset are due to label missingness and censoring.

Algorithm 1 SMS-DKL

Hyperparameters: Max BO iterations N, max training iterations M, and acquisition function a_f
Input: Sequence dataset $\mathcal{D}$
Initialize $\mathcal{A}_t$, $t \in \{1, ..., T\}$ with random samples
for $n = 1$ **to** N **do**
  for $m = 1$ **to** M **do**
    Update $\boldsymbol{\Theta}_t$, $t \in \{1, ..., T\}$ jointly
      by optimizing (5)
  end for
  Update $a_{f,t}(\mathbf{x}_t|\mathcal{A}_t)$, $t \in \{1, ..., T\}$ using (7)
  Solve $\mathbf{x}_t^* = \arg\max_{\mathbf{x}_t \in \mathbb{X}} a_{f,t}(\mathbf{x}_t|\mathcal{A}_t)$, $t \in \{1, ..., T\}$
  Sample $\mathbf{x}^*$ from $\{\mathbf{x}_t^* : t \in \{1, ..., T\}\}$
    via policy in (9)
  $\mathcal{A}_t \leftarrow \mathcal{A}_t \cup (\mathbf{x}^*, y_t^*)$, $t \in \{1, ..., T\}$
end for
Output: For each $t \in \{1, ..., T\}$, the tuple $(\mathbf{x}_t, y_t)$ with the best value of y_t in $\mathcal{A}_t$