Online Multiple Kernel Regression

Doyen Sahoo
School of Information Systems
Singapore Management University
Singapore
doyensahoo@gmail.com

Steven C.H. Hoi
School of Information Systems
Singapore Management University
Singapore
stevenhoi@gmail.com

Bin Li
Economics and Management School
Wuhan University
P. R. China
binli.whu@whu.edu.cn

ABSTRACT

Kernel-based regression represents an important family of learning techniques for solving challenging regression tasks with non-linear patterns. Despite being studied extensively, most of the existing work suffers from two major drawbacks: (i) they are often designed for solving regression tasks in a batch learning setting, making them not only computationally inefficient and but also poorly scalable in real-world applications where data arrives sequentially; and (ii) they usually assume a fixed kernel function is given prior to the learning task, which could result in poor performance if the chosen kernel is inappropriate. To overcome these drawbacks, this paper presents a novel scheme of Online Multiple Kernel Regression (OMKR), which sequentially learns the kernel-based regressor in an online and scalable fashion, and dynamically explore a pool of multiple diverse kernels to avoid suffering from a single fixed poor kernel so as to remedy the drawback of manual/heuristic kernel selection. The OMKR problem is more challenging than regular kernel-based regression tasks since we have to on-the-fly determine both the optimal kernel-based regressor for each individual kernel and the best combination of the multiple kernel regressors. In this paper, we propose a family of OMKR algorithms for regression and discuss their application to time series prediction tasks. We also analyze the theoretical bounds of the proposed OMKR method and conduct extensive experiments to evaluate its empirical performance on both real-world regression and times series prediction tasks.

Categories and Subject Descriptors

I.2.6 [**Artificial Intelligence**]: Learning

General Terms

Theory, Algorithms, Experimentation

Keywords

Online Learning; Multiple Kernel Learning; Kernel Regression; Time Series Prediction

1. INTRODUCTION

Kernel methods have been extensively studied for regression tasks and found successes in many real-world applications [29, 28]. In contrast to linear regression methods, kernel-based regression methods are able to tackle challenging non-linear regression tasks using the kernel trick that implicitly maps data from the original space to a high or even infinite dimensional space by means of a kernel function. Although a variety of kernel methods have been proposed for regression tasks [28], most conventional kernel methods suffer from two major drawbacks. First of all, they are often designed for solving regression tasks in a batch learning setting. This often results in a high re-training cost when there is any new training data, making them poorly scalable in many real-world applications where data arrives sequentially. Second, they usually assume that prior to the learning task, a fixed kernel function is given either by manual selection or via cross validation. This could result in poor performance if the chosen kernel is inappropriate in a new environment, which happens commonly for some real-world applications, such as time series prediction where data observations can be non-stationary and the optimal kernel function may change over time.

To overcome the above drawbacks, this paper investigates a novel scheme of Online Multiple Kernel Regression (OMKR), which sequentially learns a kernel-based regressor with multiple kernels in an online fashion for regression tasks. On one hand, the proposed OMKR technique, as an online learning method that often makes simple incremental update for a new training data example, avoids the expensive re-training cost of conventional batch kernel methods, and thus significantly improves the efficiency and scalability, especially when handling data stream applications. On the other hand, OMKR explores a pool of multiple diverse kernels to remedy the drawback of using a single fixed kernel by existing kernel-based regression methods that often suffer considerably when the single kernel is inappropriate.

The proposed OMKR problem is however very challenging since we not only need to sequentially learn the optimal kernel-based regressor for each individual kernel in the pool, but also need to simultaneously decide the best way of combining the multiple kernel regressors on the fly at every learning round. We tackle the challenges by (i) exploring two online kernel regression algorithms, Widrow-Hoff learning [33] and NORMA learning [17], for online regression tasks with each individual kernel; and (ii) determining the best combination of the multiple kernel regressors by applying two online learning techniques: *Hedge* algorithm [9] that can track the best kernel regressor, and *Online Gradient Descent*(OGD)[38] that can find the optimal linear combination. We analyze the theoretical bound of OMKR, and also discuss a natural extension of OMKR for the prediction of Autoregressive (AR) time series. To validate the efficacy of the proposed method, we conduct extensive experiments by evaluating the proposed algorithms on both real-world regression and time series datasets, in which our empirical results show that OMKR significantly outperforms conventional single kernel online regression approaches for most cases, especially for time series prediction tasks.

The rest of the paper is organized as follows. Section 2 reviews the related work of both online learning and kernel methods. Section 3 gives the problem formulation of online learning with multiple kernels, and then presents the OMKR algorithms followed by theoretical analysis. Section 4 presents our experimental results and discussions, and finally Section 5 concludes this paper.

2. RELATED WORK

In this section, we review some of major related work in online learning and kernel learning in the context of OMKR.

2.1 Online Learning

Online learning algorithms have been extensively explored in different contexts and applications [6, 35, 22]. For more references please refer to [27, 5, 14]. Many online algorithms have been proposed for extending kernel methods in an online setting, in which several techniques have have been proposed for online kernel regression, such as Naive Online R_{reg} Minimization Algorithm (NORMA) [17], Online Passive Aggressive Regression [6], Sparse Implicit Online Learning with Kernels (ILK and SILK) [26] and Primal Online Algorithm (PRIONA) [3].

In addition, some kernel-based online learning studies focus on the budget issue [7, 4]. These help to speed up computation cost by bounding the number of support vectors. Some well-known example algorithms include Forgetron [8], Projectron [23], and the Bounded Online Gradient Descent (BOGD) [36]. Further, our work is also related to online prediction with expert advice [9, 20, 32]. One of the most well-known algorithms is the Hedge Algorithm [9], which was a direct generalization of Weighted Majority Algorithm [20].

2.2 Kernel Learning

Most kernel methods often assume that a predefined parametric kernel is given a priori, where the parameters are chosen either manually or via cross validation. Kernel learning aims to learn an effective kernel from data automatically. Some studies have attempted to learn kernel functions or matrices from labeled and unlabeled data. Examples include marginalized kernels [16], idealized kernel learning [18], graph-based spectral kernel learning [2, 13], and non-parametric kernel learning [11, 37]. These methods often follow a batch (and transductive) learning setting and thus are difficult to be applied in an online learning scenario.

Another prevalent kernel learning technique is Multiple Kernel Learning (MKL) [19], which aims to find the optimal combination of multiple kernels. Unlike most existing MKL techniques that are batch learning [19, 30, 10], our work focuses on online regression tasks, and is related to existing online MKL studies that focus on classification tasks [15, 12] and that addresses structured prediction [21].

3. ONLINE MULTIPLE KERNEL REGRESSION

3.1 Overview

In this section, we present the proposed Online Multiple Kernel Regression (OMKR) scheme. We will first motivate the problem by introducing the formulation of batch Multiple Kernel Learning (MKL). We then present our OMKR framework, the detailed algorithms for addressing different challenges, and finally theoretical analysis of OMKR.

Consider a set of training examples $\mathcal{D} = (\mathbf{x}_i, y_i), i = 1, \dots, T$ where $\mathbf{x}_i \in \mathbb{R}^d$, $y_i \in \mathbb{R}$ and a collection of m kernel functions $\mathcal{K} = \{\kappa_i : \ \chi \times \chi \to \mathbb{R}, i = 1, \dots, m\}$. Multiple Kernel Learning aims to learn a kernel-based prediction model by identifying the best linear combination of the m kernels, that is, a weighted combination $\theta = (\theta_1, \dots, \theta_m)$. The learning task can be cast into the following optimization [19]:

$$\min_{\theta \in \Delta} \min_{f \in \mathcal{H}_K(\theta)} \frac{1}{2}|f|^2_{\mathcal{H}_{K(\theta)}} + C\sum_{i=1}^{n} \ell(f(\mathbf{x}_i), y_i) \tag{1}$$

where $\Delta = \{\theta \in \mathbb{R}^m_+ | \theta^T \mathbf{1}_m = 1\}$, $K(\theta)(\cdot,\cdot) = \sum_{i=1}^{T} \theta_i \kappa_i(\cdot,\cdot)$ and $\ell(f(\mathbf{x}_i), y_i)$ is a convex loss function.

The above convex optimization problem of regular batch MKL can be solved by different schemes [30, 34, 10]. Despite being studied extensively, it remains very challenging when solving the batch MKL for large-scale applications. Besides, similar to most batch kernel methods, regular MKL has some drawbacks: (i) the trained model, if it is not re-trained with new data, may work poorly for non-stationary data in a new environment; but (ii) the re-training cost is extremely expensive for data streams, making it non-scalable.

3.2 OMKR Framework

To overcome the limitations of MKL for a regression task, we propose a new scheme of Online Multiple Kernel Regression (OMKR) by applying the emerging online multiple kernel learning principle [12] for tackling regression tasks, which attempts to sequentially learn the online multiple-kernel regressor given a new data example using a two-step updating scheme: (i) update the set of kernel-based regressors for each individual kernel; and (ii) update the weights for combining the multiple kernel regressors. In the following, we discuss the details of the proposed algorithms for tackling online regression tasks at each of the two steps.

3.2.1 Learning Online Kernel-Based Regressors

The goal of this task is to learn a regression function $f_t \in \mathcal{H}_\kappa$ in an online setting, where $\mathcal{H}_\kappa$ a reproducing kernel Hilbert space (RKHS) induced by a given specific kernel $\kappa \in \mathcal{K}$. We solve this task by exploring two online regression solutions: Kernel Widrow-Hoff [33] and NORMA [17], which follows the same principle of Online Gradient Descent (OGD) [38] for online convex optimization and but optimizes two slightly different objective functions.

Kernel Widrow-Hoff Learning. Given a sequence of data instances $\mathcal{D} = (\mathbf{x}_i, y_i), i = 1, \dots, T$, the goal of kernelized Widrow-Hoff learning is to minimize the total cumulative loss over the whole regression task $\mathcal{L}$ defined as follows:

$$\mathcal{L} = \Sigma_{t=1}^T \ell(f_t(x_t), y_t) \triangleq \Sigma_{t=1}^T \mathcal{L}_t(f_t) \tag{2}$$

where $f_t(x_t)$ is the prediction made by a kernel regressor on the t-th instance, $\ell(f_t(\mathbf{x}_t), y_t)$ denoted by $\mathcal{L}_t(f_t)$ for short, is a convex loss function. Following OGD [38], we have the following online update rule given a data instance $(\mathbf{x}_t, y_t)$:

$$f_{t+1} \leftarrow f_t - \eta_t \nabla \mathcal{L}_t(f_t) \tag{3}$$

where $\eta_t > 0$ is a learning rate parameter that can be either a small constant $\eta_t = \eta$ used in Widrow-Hoff [33] or a factor depending on t. When choosing the squared loss for ℓ:

$$\ell(f_t(\mathbf{x}_t), y_t) = (f_t(\mathbf{x}_t) - y_t)^2,$$

we have the online updating rule expressed explicitly as

$$f_{t+1}(\cdot) \leftarrow f_t(\cdot) - \eta_t(f_t(\mathbf{x}_t) - y_t)\kappa(\mathbf{x}_t, \cdot). \tag{4}$$

NORMA. The above method has two potential drawbacks. First, it may lead to overfitting when dealing with noisy data. Second, due to the use of squared loss, almost every training instance will be added as support vectors (unless $f_t(x_t)$ is identical to y_t), making the prediction function computationally intensive when handling large-scale datasets. To overcome these drawbacks, we explore another online regression scheme by following the idea of NORMA [17], which replaces $\mathcal{L}_t(f_t)$ by the following regularized loss:

$$\mathcal{L}_t(f_t) = \frac{\lambda}{2}||f||^2_{\mathcal{H}_\kappa} + \ell(f_t(\mathbf{x}_t), y_t) \tag{5}$$

By the OGD principle, we have the online updating rule as:

$$f_{t+1} \leftarrow (1 - \eta_t\lambda)f_t - \eta_t \nabla \ell(f_t(\mathbf{x}_t), y_t) \tag{6}$$

where $\eta_t > 0$ is the learning rate parameter. Instead of using the square loss, we exploit the ϵ-insensitive loss function which is defined as

$$\ell(f_t(\mathbf{x}_t), y_t) = \max(0, |y_t - f_t(\mathbf{x}_t)| - \epsilon),$$

where ϵ represents the width of the insensitivity zone. We can further modify the loss function by making ϵ as a variable of the optimization:

$$\ell_t(f_t, \mathbf{x}_t) = \max(0, |y_t - f_t(\mathbf{x}_t)| - \epsilon) + \nu\epsilon_t \tag{7}$$

where $\nu > 0$ is a parameter, and ϵ_t is a variable to be updated in online learning process. Using the above loss function, we can derive the online updating rule for NORMA:

$$f_{t+1} \leftarrow \begin{cases} (1 - \eta_t\lambda)f_t + \eta_t * sgn(d)\kappa(\mathbf{x}_t, \cdot) & \text{if } |d| > \epsilon_t \\ (1 - \eta_t\lambda)f_t & \text{otherwise} \end{cases} \tag{8}$$

$$\epsilon_{t+1} \leftarrow \begin{cases} \epsilon_t + (1 - \nu)\eta_t & \text{if } |d| > \epsilon_t \\ \epsilon_t - \eta_t\nu & \text{otherwise} \end{cases} \tag{9}$$

where we denote $d = y_t - f(\mathbf{x}_t)$.

Remark. For both of the above methods, at the end of each online learning round, we can express the prediction function of the regressor as a kernel expansion [25]:

$$f_{t+1}(\mathbf{x}) = \Sigma_{i=1}^t \alpha_i \kappa(\mathbf{x}_i, \mathbf{x})$$

where the α_i coefficients are computed based on the updating rules in (4) or (8). When $\alpha_i \neq 0$, the i-th instance is often called as a Support Vector (SV). Thus, the time complexity for prediction is linear with respect to the number of SV's. When using the squared loss, we will have $\alpha_i \neq 0$ for almost every instance, leading to a large number of support vectors. By contrast, when using the ϵ-insensitive loss, whenever the difference between the prediction on the i-th instance $f_i(x_i)$ and y_i is small enough, i.e., within the ϵ tube, we have $\alpha_i = 0$, which thus generates a much smaller SV size and significantly improves the prediction efficiency.

3.2.2 Learning the Best Kernel Combination

The previous online kernel regression method allows us to learn a set of kernel regressors $f_t^i \in \mathcal{H}_{\kappa_i}, i = 1, \ldots, m$ with respect to the pool of multiple diverse kernels $\mathcal{K}$. The idea of OMKR is to learn an effective regressor $F_t(\mathbf{x})$ by combining the set of multiple kernel regressors:

$$F_t(\mathbf{x}) = \sum_{i=1}^m w_t^i \mathbf{f}_t^i(\mathbf{x}) \tag{10}$$

where $w_t^i \in \mathbb{R}$ denotes the combination weight for the i-th kernel regressor. The remaining problem then is to determine the appropriate combination weights $\mathbf{w}_t$ for the set of kernels. We note that this is a very challenging task since we may not have prior knowledge for empirical performance of each kernel, and the optimal combination weights may even change over time in the online learning process especially when dealing with non-stationary data.

One naive solution is to simply adopt a uniform combination for all the kernels, i.e., $w_t^i = 1/m$, which does explore all the kernels, but often results in sub-optimal performance, as observed in our empirical studies. In this section, we attempt to learn the best kernel combination weights by exploring two different online learning algorithms: the Hedge algorithm [9] and the OGD algorithm [38]. We will first present each algorithm in detail and finally discuss their strengths and weaknesses for different scenarios.

Hedge Algorithm: The Hedge algorithm is the most popular online algorithm for solving the problem of decision-theoretic online learning or known as prediction with expert advice [32, 5]. Specifically, by treating each online kernel regressor as an expert, the Hedge algorithm aims to minimize the regret of the learner for the regression task, which is the difference between the learner's cumulative loss and the cumulative loss of the best kernel regressor. In theory, Hedge can achieve an optimal upper bound of regret $O(T \ln m)$ with T learning rounds and m kernel regressor experts. It is thus an ideal online learning algorithm for tracking the best online kernel regressor especially when there is some kernel regressor significantly dominates the rest.

Specifically, the Hedge algorithm runs in a fairly simple way. Consider the OMKR problem, at the beginning, the combination weights $\mathbf{w}_t$ are initialized as a uniform distribution, i.e., $w_1^i = 1/m, i = 1, \ldots, m$. At the end of each learning round, according to the performance of the multiple kernel regressors, the weights are updated by:

$$w_{t+1}^i = w_t^i \beta^{\ell_t^i}, i = 1, \ldots, m \tag{11}$$

where $\beta \in (0, 1)$ is a discounting (learning rate) parameter, and ℓ_t^i denotes the loss suffered by the i-th kernel regressor at round t, i.e., $\ell_t^i = \ell(f_t^i(x_t), y_t)$. Finally, we normalize all w_{t+1}^i's to ensure the combination weights as a distribution.

We refer to the proposed OMKR algorithm that adopts the Hedge algorithm as the *Deterministic OMKR* (Hedge) algorithm, as shown in Algorithm 1. In the algorithm, we can update each kernel regressor f_{t+1}^i by adopting either the Widrow-Hoff learning in (4) or NORMA in (8).

Although Hedge is ideal for tracking the best kernel regressor, it is not always perfect for solving a practical OMKR problem since our goal is to learn the best combination of multiple kernels. In the following, we present an online gradient descent (OGD) based algorithm that attempts to learn the optimal linear combination of multiple kernel regressors.

Algorithm 1 Deterministic OMKR (Hedge)

```
INPUT:
  - Kernels: $\kappa(\cdot,\cdot) : \chi \times \chi \to i = 1, \ldots, m$
  - Discounting Parameter: $\beta \in (0, 1)$
  - Step size parameter for each kernel: $\eta$
  - Regression parameters: $\lambda$ and $\nu$ for OMKR(NORMA)
Initialization: $\mathbf{f}_1 = \mathbf{0}$, $\mathbf{w}_1 = \frac{1}{m}\mathbf{1}$
  for t = 1,...,T do
    Receive instance: $x_t$
    Predict $\hat{y}_t = \sum_{i=1}^{m} w_t^i f_t^i(x_t)$
    Reveal true value $y_t$
    for i = 1,...,m do
      Set $\ell_t^i = \ell(f_t^i(x_t), y_t)$;
      Update $f_{t+1}^i$ = Eq. (4) OR (8)
      Update $w_{t+1}^i = w_t^i \beta^{\ell_t^{*i}}$ where $\ell_t^{*i} = (f_t^i(x_t) - y_t)^2$;
    end for
    Set $w_{t+1}^i = \frac{w_t^i}{W_t}$ where $W_t = \sum_{i=1}^{m} w_t^i,\ i = 1, \ldots, m$
  end for
```

OGD Algorithm: Our goal is to learn the optimal combination weight vector $\mathbf{w}_t \in \mathbb{R}^m$ for combining the multiple kernel regressors. It can be cast into the following online optimization

$$\mathbf{w}_{t+1} \leftarrow \arg\min_{\mathbf{w}} \ell(\mathbf{w}^\top \mathbf{f}_t(\mathbf{x}_t), y_t) \triangleq (\mathbf{w}^\top \mathbf{f}_t(\mathbf{x}_t) - y_t)^2 \quad (12)$$

where $\mathbf{f}_t(\mathbf{x}_t)$ is a vector representing the predictions made by all the kernel regressors on instance $\mathbf{x}_t$, and ℓ is a loss function denoting the loss suffered by the OMKR. We simply adopt the squared loss in our solution (though it may also include a regularizer). Following the OGD, we can derive the updating rule as follows:

$$\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t - \eta_w(\hat{y}_t - y_t)\mathbf{f}_t(\mathbf{x}_t) \quad (13)$$

where η_w is a learning rate parameter, and $\hat{y}_t = \mathbf{w}^\top \mathbf{f}_t(\mathbf{x}_t)$.

Using the above OGD algorithm for learning the optimal combination weights, we propose another OMKR scheme, called *Deterministic OMKR*(OGD), as shown in Algorithm 2. Like in OMKR(Hedge), we can also update each kernel regressor by either Widrow-Hoff in (4) or NORMA in (8).

Algorithm 2 Deterministic OMKR (OGD)

```
INPUT:
  - Kernels: $\kappa(\cdot,\cdot) : \chi \times \chi \to i = 1, \ldots, m$
  - Learning rate parameter: $\eta_w$
  - Step size parameter for each kernel: $\eta$
  - Regression parameters: $\lambda$ and $\nu$ for OMKR(NORMA)
Initialization: $\mathbf{f}_1 = \mathbf{0}$, $\mathbf{w}_1 = \mathbf{0}$
  for t = 1,...,T do
    Receive instance: $x_t$
    Predict $\hat{y}_t = \sum_{i=1}^{m} w_t^i f_t^i(x_t)$
    Reveal true value $y_t$
    for i = 1,...,m do
      Set $\ell_t^i = \ell(f_t^i(x_t), y_t)$;
      Update $f_{t+1}^i$ = Eq. (4) OR (8)
    end for
    Update $\mathbf{w}_{t+1} = \mathbf{w}_t - \eta_w(\hat{y}_t - y_t) f_t(x_t)$
  end for
```

Remark. In online MKL work related to classification [15, 12], Hedge algorithm was used to combine multiple predictions. In contrast, our proposed OGD approach interprets the kernel predictions as new rich features which can be combined linearly. In terms of update rules, Hedge makes multiplicative updates while OGD makes additive updates. Further, for the combination weight vector $\mathbf{w}_t$, Hedge always keep $\mathbf{w}_t$ a distribution ($w_t^i \geq 0$ and $\sum_i w_t^i = 1$) while OGD is able to learn any real-valued vector for $\mathbf{w}_t$. In general, both Hedge and OGD have their different merits. Hedge is good at tracking the best kernel regressor, while OGD is good at learning the optimal combination of multiple kernel regressors. However, OGD often suffers from slow convergence rate. In practice, the empirical performance of OMKR(Hedge) and OMKR(OGD) may vary a lot in different scenarios. Due to the nature of multiplicative update of Hedge, it converges quickly, and in an online setting, may tend to achieve better performance than OGD, if the dataset is small, or if the pattern changes due to non-stationarity. We conduct more in-depth analysis through our extensive experimental studies in Section 4.

3.3 Application to Time Series Prediction

OMKR can be applied a variety of online regression tasks, especially for mining data streams. A natural application of OMKR is Time Series Prediction, which is the task of predicting the future value based on given past values. Kernel methods have been commonly used for solving such problems [31, 24]. We first introduce the popular time series prediction model known as Autoregressive (AR) model, and then present a kernelized AR model, followed by showing the application of OMKR for time series prediction.

Autoregressive(AR) model is used for a univariate time series where the value of the series at a particular time is linearly dependent on its own previous values. An $AR(p)$ model denotes an autoregressive process of order p, i.e., y_t is described by a noisy linear combination of $[y_{t-1}y_{t-2}\ldots y_{t-p}]$:

$$y_t = c + \Sigma_{i=1}^{p} \zeta_i y_{t-i} + \epsilon_t \quad (14)$$

where c is a constant, ϵ_t is white noise, and ζ_i are the parameters describing the dependency. A simple $AR(p)$ model assumes linear dependency on the previous p values. This may not be true. We use kernels to explore nonlinear dependencies. The kernelized $AR(p)$ model is given by:

$$y_t = c + f(y_{t-1}, y_{t-2}, \ldots, y_{t-p}) + \epsilon_t = c + f(Y_{t-1}^p) + \epsilon_t$$

where $f(Y_{t-1}^p) \in \mathcal{H}_\kappa$ is the prediction of the regression function using a kernel κ. Here, a new challenge arises, i.e., to choose the appropriate kernel function. In addition, another issue is how to choose the appropriate parameter p.

To solve these two issues, we propose to construct a pool of multiple kernels for varying values of parameter p. For example, for $p \in [p_1, p_2, \ldots, p_k]$, and m kinds of diverse kernels, we can create the following pool of mk kernel functions:

$$\mathcal{K} = \left\{\kappa^i(Y_t^{p_1}, \cdot), \ldots, \kappa^i(Y_t^{p_k}, \cdot) \text{ for } i = 1, \ldots, m.\right\}$$

The above can now be directly plugged into the OMKR framework for solving time series prediction tasks. In comparison to existing kernel methods for times series prediction, the proposed OMKR solution enjoys the important advantages of avoiding tedious kernel selection and parameter selection (e.g., p) and exploiting the power of combining multiple kernels for more accurate prediction.

3.4 Theoretical Analysis

Without loss of generality we assume that $\forall\ i,\ \forall\ t,\ \kappa^i(x_t \cdot x_t) \leq 1$, and $\ell_t(f_t^i(\mathbf{x}_t), y_t) \leq 1$ We define the optimal kernel regressor with respect to the squared loss (Widrow-Hoff) as:

$$F(\kappa_i, \ell, \mathcal{D}) = \min_{f \in \mathcal{H}_\kappa} \left[\frac{\Sigma_{t=1}^T (f(x_t) - y_t)^2}{1-\eta} + \frac{||f||^2}{\eta} \right] \quad (15)$$

where $\mathcal{D}$ is a sequence of instances.

THEOREM 1. *After receiving a sequence of T instances, the cumulative loss suffered by OMKR (Hedge) using the Widrow-Hoff Algorithm is bounded as*

$$\mathcal{L}_{OMKR} \leq \frac{\ln(\frac{1}{\beta})}{1-\beta} \min_{1 \leq i \leq m} F(\kappa_i, \ell, \mathcal{D}) + \frac{\ln(m)}{1-\beta} \quad (16)$$

Here $\mathcal{L}_{OMKR}$ is the total loss suffered at each prediction, and due to the convexity of the loss function, we have

$$\mathcal{L}_{OMKR} = \Sigma_{t=1}^T \ell(\Sigma_{i=1}^m w_t^i f_t^i(x_t), y_t) \leq \Sigma_{t=1}^T \Sigma_{i=1}^m w_t^i \ell(f_t^i(x_t), y_t)$$

and by choosing $\beta = \frac{\sqrt{T}}{\sqrt{T}+\sqrt{\ln m}}$, we get:

$$\mathcal{L}_{OMKR} \leq (1 + \sqrt{\frac{\ln m}{T}} \min_{1 \leq i \leq m} F(\kappa_i, \ell, \mathcal{D}) + \ln m + \sqrt{T \ln m})$$

PROOF. The proof follows from combining the proof of Hedge Algorithm and the Widrow-Hoff Regression. Let $\phi_t^i = ||f_t^i - f||_2^2$ for any $f \in \mathcal{H}_{\kappa_i}$. Also, let Δ_t denote the change in f during each update, such that $\Delta_t = \eta(f_t(x_t) - y_t)\kappa(x_t, \cdot)$. We also define $\ell_t = f_t(x_t) - y_t$ as the signed error suffered by f_t, and $\ell_t^* = f(x_t) - y_t$ be the signed error suffered by f.

$$\begin{aligned} \phi_{t+1}^i - \phi_t^i &= ||f_{t+1}^i - f||_2^2 - ||f_t^i - f||_2^2 \\ &= ||\Delta_t||_2^2 - 2(f_t^i - f) \cdot \Delta_t \\ &= \eta^2 \ell_t^{i2} \kappa(x_t \cdot x_t) - 2\eta\ell_t\kappa(x_t, \cdot) \cdot (f_t^i - f) \\ &\leq \eta^2 \ell_t^{i2} - 2\eta\ell_t^2 + 2\eta\ell_t\ell_t^* \\ &= \eta^2 \ell_t^{i2} - 2\eta\ell_t^{i2} + 2\eta \left[(\ell_t^i \sqrt{1-\eta})(\frac{\ell_t^*}{\sqrt{1-\eta}}) \right] \end{aligned}$$

The inequality follows from the assumption $\kappa(x_t \cdot x_t) \leq 1$.

$$\begin{aligned} \phi_{t+1}^i - \phi_t^i &\leq \eta^2 \ell_t^{i2} - 2\eta\ell_t^{i2} + \eta(1-\eta)\ell_t^{i2} + \frac{\eta}{1-\eta}\ell_t^{*2} \\ &= -\eta\ell_t^{i2} + \frac{\eta}{1-\eta}\ell_t^{*2} \end{aligned} \quad (17)$$

In the above equation we use the algebraic inequality $ab \leq (a^2 + b^2)/2$. From this, by assuming $f_1 = \mathbf{0}$, and using a telescoping sum, it is very simple to prove that

$$\mathcal{L}_{WH}^i \leq \min_{f \in \mathcal{H}_\kappa} \left[\frac{\Sigma_{t=1}^T (f(x_t) - y_t)^2}{1-\eta} + \frac{||f||^2}{\eta} \right] \quad (18)$$

where $\mathcal{L}_{WH}^i$ is the cumulative loss suffered by the regression function learnt by the the Widrow-Hoff Algorithm in the RKHS by the i^{th} kernel. Plugging this into the Hedge Algorithm gives us the bound. The choice of β maybe overestimated because of the assumption that the loss suffered by the algorithm is T. $\square$

Similarly, bounds can be derived for any generic convex loss function by using online gradient descent for learning each regression function. For a decaying $\eta_t = \frac{1}{\sqrt{t}}$, OGD can achieve a sublinear regret bound of $O(\sqrt{T})$ with respect to the best linear combination of multiple kernel predictions. For a fixed η, the bound is weaker and may indicate poor learning. However, a fixed η is more suitable in a non-stationary environment, as it can adapt to a changing pattern faster.

3.5 Stochastic and Budget OMKR

In the worst case, all the instances become support vectors for each kernel in the OMKR framework(which is invariably the case when using squared loss). For the t^{th} instance, the time taken for a prediction to be made by a single kernel is in $O(t)$, and for m predictions to be made by m kernels is in $O(mt)$. However, not all kernels are good candidates for prediction, especially when their weights are low. In addition, not all the historical instances are good candidates for making the prediction, particularly in a non-stationary setting. With this motivation, we propose stochastic update and budget online kernel learning strategies.

3.5.1 Stochastic Update for OMKR

An update to a kernel regressor involves adding a new support vector. If SVs are not added to less important kernels, the time taken for prediction by these kernels is significantly reduced. The intuition is if there is only one good kernel or a small subset of good performing kernels, it is only these should be given more data to learn the function, and the poor kernels are still allowed to make predictions (but with limited data), which takes much lesser computational time. We define a probability sampling denoted by q_t^i, which determines the probability of a kernel being selected for updates.

$$q_t^i = \frac{|w_t^i|}{\max_{1 \leq j \leq m} |w_t^j|} \quad (19)$$

This indicates that higher the absolute weight, the higher is the probability, and the best kernel has a probability of 1. When OMKR(Hedge) is used the weights can never be negative. In case of OGD updates in weights, there is a theoretical possibility for the weights to become negative, and hence we take absolute values to compute q_t^i, so as to account for weights having the maximum impact on the prediction. To prevent kernels with low weights, that do not have a significant impact to the prediction, from completely losing out, we introduce a smoothing parameter $\delta \in (0, 1)$. The idea is to add a small component of uniform weights. The new probability of a kernel being selected for update is denoted by:

$$p_t^i = (1-\delta)q_t^i + \frac{\delta}{m} \quad (20)$$

Here δ is a small value. A similar idea was used in [1], to tradeoff between exploration and exploitation. Using p we sample a subset of kernels based on Bernoulli Sampling, i.e., $m_t^i = Bernoulli(p_t^i)$. Only those kernels that are selected will be chosen for an update. The steps are described in Algorithm 3.

THEOREM 2. *After receiving a sequence of T instances the expected loss of the Stochastic Update OMKR (Hedge) denoted by $\mathbb{E}[\mathcal{L}] = \mathbb{E}[\Sigma_{t=1}^T \ell(\Sigma_{i=1}^m \theta_t^i f_t^i(x_t), y_t)]$ is bounded as:*

$$\mathbb{E}[\mathcal{L}] \leq \frac{\ln(\frac{1}{\beta})}{\delta(1-\beta)} \min_{1 \leq i \leq m} F(\kappa_i, \ell, \mathcal{D}) + \frac{\ln(m)}{\delta(1-\beta)} \quad (21)$$

Algorithm 3 Stochastic OMKR scheme

```
INPUT:
  - Kernels: $\kappa(\cdot,\cdot):\chi\times\chi\rightarrow i=1,\dots,m$
  - Update Parameter: $\beta\in(0,1)$ if Hedge or $\eta_w$ for OGD
  - Smoothing Parameter: $\delta\in(0,1)$
  - Step Size Parameter for each kernel: $\eta$
  - Regression parameters:$\lambda,\nu$ for OMKR(NORMA) NORMA)
Initialization: $\boldsymbol{f_1}=\mathbf{0}$, $\mathbf{w_1}=\frac{1}{m}\mathbf{1}$
  for t = 1,...,T do
    Receive instance: $x_t$
    Predict $\hat{y_t}$ based on Hedge or OGD combination
    Reveal true value $y_t$
    $q_t^i=\frac{|w_t^i|}{\max\limits_{1\le j\le m}|w_t^j|},\ i=1,\dots,m$
    $p_t^i=(1-\delta)q_t^i+\frac{\delta}{m},\ i=1,\dots,m$
    Sample $m_t^i=BernoulliSampling(q_t^i),\ i=1,\dots,m$
    for i = 1,...,m do
      Set $\ell_t^i=\ell(f_t^i(x_t),y_t)$
      if $m_t^i==1$ then
        Update $f_{t+1}^i=$ Eq. (3) OR (6)
      end if
    end for
    Update $\mathbf{w}_{t+1}$ based on Hedge or OGD
  end for
```

and by choosing $\beta=\frac{\sqrt{T}}{\sqrt{T}+\sqrt{\ln m}}$, *we get:*

$$\mathbb{E}[\mathcal{L}]\le\frac{1}{\delta}(1+\sqrt{\frac{\ln m}{T}}\min_{1\le i\le m}F(\kappa_i,\ell,\mathcal{D})+\ln m+\sqrt{T\ln m})$$

The proof is similar to the proof of OMKR(Hedge). We omit the details.

3.5.2 Budget OMKR

Often, a subset of instances can explain the data as well as the entire data. Further, in a non-stationary time series setting, it is common to introduce a sliding window so as to give importance to only the most recent instances. In our case, we have a sliding window of the most recent support vectors that explain the data. This is also particularly helpful in the case of NORMA, where in each iteration, the old support vectors get reduced by a factor of $(1-\eta\lambda)$. As t grows, the α values of the old support vectors get reduced to almost zero. Such support vectors can be ignored without any significant impact to the prediction. Therefore, we propose a parameter τ which restricts the total number of support vectors that are allowed to be stored by each regressor. The older support vectors are deleted.

3.5.3 Comparison between Stochastic and Budget

Stochastic OMKR improves efficiency by reducing the number of support vectors of poor performing kernels. However, if all kernels are equally good, there is no reduction in computational time. The budget strategy restricts the number of support vectors of each kernel, and has a worst case complexity of $O(m\tau)$ in each iteration to make the prediction. Both address non-stationarity in different ways. Stochastic approach trades off between exploration and exploitation to determine if another kernel has become more suitable for the changing pattern, and the budget approach gives more importance to the most recent data.

4. EXPERIMENTAL EVALUATION

In this section, we empirically evaluate the performance of all OMKR variants on regression data and time series data.

4.1 Experimental Setup

4.1.1 Data

We use five regular regression datasets and seven time series datasets. The data is from different applications, with a wide range of data size and dimensionality. All data attributes including the target were scaled to $[0,1]$. The algorithms were run on ten random permutations of the regular regression datasets to establish robustness. Such permutations are not applicable in the case of time series. The details of the datasets used can be seen in Table 1.

Table 1: List of Datasets

ID	Name	# Instances	# Attributes
	Regression Datasets		
D1	Abalone	4177	8
D2	Parkinsons	5875	20
D3	Spacega	3107	6
D4	Cadata	20640	8
D5	Add10	9792	11
	Time Series Datasets		
D6	Laser	10073	20|10
D7	Physiological	17000	2
D8	Currency Exch. 1	3000	20|10
D9	Currency Exch. 2	3000	20|10
D10	Astrophysical	598	20|10
	Large Time Series Datasets		
D11	Santafe Computer	100000	20|10
D12	Twitter	583250	77

Datasets D1, D2 and D12 were taken from the UCI repository[1], D3-D4 from StatLib[2], D5 is a synthetic dataset obtained from Delve[3]. D6-D11 are datasets from the Santa Fe Time Series Competition Data[4]. D6 is stationary, D7 is non-stationary, and unlike other time series data, is not univariate, but is dependent on 2 attributes, D8 and D9's stationarity property is unknown, and D10 is characterized by noise. D11 is non-stationary with a slow drift. D12 is about predicting buzz in social media. For univariate time series data the attribute column having 20|10 indicates the choice of 2 kernelized $AR(p)$ process with $p=10,20$ each having its own m kernel functions.

4.1.2 Kernels

We evaluate the performance of OMKR by using a pool of 24 predefined kernels. These include 4 polynomial kernels $\kappa(x,y)=(x^Ty)^p$ of degree parameter $p=1,2,3,4$, 13 RBF kernels ($\kappa(x,y)=e^{(\frac{-||x-y||^2}{2\sigma^2})}$) of kernel width parameter σ in $[2^{-6},2^{-5},\dots,2^6]$, 5 Cauchy kernels ($\kappa(x,y)=\frac{1}{1+\frac{||x-y||^2}{\sigma^2}}$) with parameter σ in $[2^{-2},2^{-1},\dots,2^2]$, one sigmoid kernel($\kappa(x,y)=\tanh(xy)$) and a Chi-Square Kernel

[1]`http://archive.ics.uci.edu/ml/`
[2]`http://lib.stat.cmu.edu/`
[3]`http://www.cs.toronto.edu/~delve/data/datasets.html`
[4]`http://www-psych.stanford.edu/~andreas/Time-Series/SantaFe.html`

($\kappa(x, y) = 1 - \Sigma_{i=1}^{n} \frac{(x_i - y_i)^2}{\frac{1}{2}(x_i + y_i)}$). Since all our data is scaled to $[0, 1]$, we clip the kernel prediction to this range, i.e., $\hat{y_t} = \max(0, \min(1, \hat{y_t}))$.

4.1.3 *Comparison and Performance Metrics*

We compare the algorithms based on Mean Squared Error (MSE), time taken, and the weight distribution. The algorithms compared are - (i) *Regression(V)*: Best Kernel by validation; (ii) *Regression(H)*: Best Kernel in hindsight; (iii) *Uniform OMKR*: Uniform weight distribution over kernels (to see if this can eliminate the impact of a poor kernel choice); (iv) *Deterministic OMKR (Hedge)*; and (v) *Deterministic OMKR (OGD)*. We then analyze the performance of efficiency enhancing variants of OMKR and study the tradeoff between accuracy and efficiency. For the large datasets (D11 and D12), we compare only the budget versions of all algorithms.

4.1.4 *Parameter Setting*

All parameters for the regression tasks (if any), and the best kernel for Regression(V) were chosen by online validation technique. We performed a grid search and evaluated the performance of the parameters on the of first 100 instances or first 10% of the instances, whichever was lesser. The value of Hedge parameter β was fixed to 0.5 in all cases, and the learning rate η_w was fixed to 0.025 for OGD update of weights). We also conducted sensitivity analysis for the weight update parameters. The learning rate η for each kernel regression was fixed at 0.1. Since η is the same for both single kernel and multi kernel versions, its choice does not affect the comparison between Single Kernel Regression and OMKR. For budget strategies, we fixed the budget size $\tau = 500$ support vectors. In stochastic OMKR, the smoothing parameter δ was set to 0.05 in all cases.

4.2 Experimental Results and Analysis

4.2.1 *Evaluation of Deterministic OMKR*

The detailed results of single kernel regression against OMKR can be seen in Table 2. Columns Reg(V) and Reg(H) represent single kernel regression by validation and in hindsight. Columns Uniform, Hedge, OGD represent OMKR with uniform weights, weight updated by Hedge, and weight updated by OGD respectively.

With almost no exception both our proposed methods OMKR (Hedge and OGD) outperform Reg(V) very significantly, at times achieving as low as 1% of error of Reg(V). We should note that in a real world setting, it is hard to choose a better kernel for unseen data than by a validation method. Reg(H) is the best kernel in hindsight, and is not known prior to running the experiments. Despite this, OMKR algorithms significantly outperform Reg(H) in most cases. In cases, where it OMKR does not beat Reg(H), their performance is very closely matched. Thus, without any a priori knowledge, OMKR is able to outperform even the best kernel in hindsight. This is because OMKR is able to identify a linear combination of kernels, which provide complementary information to each other in order to give a weighted prediction which beats any single best kernel. Uniform OMKR is affected by the usage of certain poor kernels and its performance is very inconsistent across datasets. It never beats OMKR(OGD), and beats OMKR(Hedge) in only one case (D1-Norma). This however is probably an exception, in which the optimal linear combination is close to a uniform distribution, because of which uniform weights are probably just a lucky guess. The difference in performance by Reg(V) and Reg(H), and the poor performance by Uniform(OMKR) highlight the difficulty of choosing the best kernel function for a given task. In terms of efficiency, Deterministic OMKR takes roughly m times the amount of time take by single kernel regression.

Hedge and OGD are suitable in different scenarios. Due to a multiplicative update, Hedge converges very quickly, by identifying the single kernel that best represents the data, which is often the case. However, since Hedge only offers a linear combination of the best kernel(s), we expect the optimal linear combination determined by OGD to outperform Hedge. This does not happen if the the data is not large enough for OGD to converge to optimal linear combination, or the data is non-stationary such that the appropriate kernel function changes too frequently for OGD to be able to learn the optimal combination. We plot the cumulative mean squared error against time for some representative datasets in Figures 1 and 2. It can be seen, that in most cases, OMKR(Hedge) attains a very low MSE from the beginning and does not improve much further, whereas, OMKR(OGD) starts with a relatively higher MSE, but it is continuously improving its performance. Referring back to Table 2, it can be seen that in general that OMKR(OGD) has relative advantage in larger datasets, and OMKR(Hedge) in smaller ones. Additionally, we also look at the weight distribution attained by the algorithms, which is shown in Figure 3. The weight distribution by OMKR(Hedge) concentrates largely on the best kernel in hindsight, and otherwise has weights over certain reasonably good performing kernels. Unlike OMKR(Hedge), OMKR(OGD) does not have a concentrated distribution of weights over few kernels.

4.2.2 *Evaluation of OMKR Efficiency Enhancers*

The MSE and the time taken by Deterministic, Stochastic and Budget OMKR are detailed in Tables 3 and 4. Clearly the time taken by both stochastic and budget techniques is significantly lower than Deterministic OMKR. Despite this, in most cases, the efficiency enhancers give comparable MSEs with respect to Deterministic OMKR. In many cases, particularly time series, the variants are able to outperform the deterministic version. This shows their ability to retain important information from the data, and adapt to changes in the pattern. Stochastic is faster than budget in smaller datasets, but in larger datasets, the number of SVs in stochastic start dominating even if only for a few kernels, and hence Budget is faster.

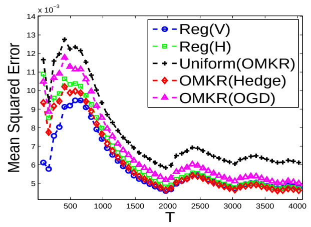
Using Budget OMKR, we run the algorithm on two large datasets (D11 and D12). In this case, we use a budget for all 5 algorithms. The details are shown in Table 5. The results are consistent with our previous findings of OMKR outperforming single kernel regression.

4.2.3 *Effect of weight update parameters β and η_w*

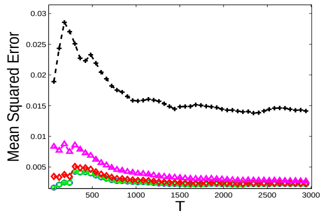
OMKR(Hedge) is not very sensitive to the value of the discount rate parameter β. There is a reasonably large range of values of β in which OMKR(Hedge)'s relative performance to other algorithms remains the same. OMKR(OGD)'s sensitivity to the learning rate η_w shows a tradeoff between large and small learning rates. This behavior is typical of all gradient descent algorithms.

Table 2: Single Kernel Regression vs Multiple Kernel Regression:
Each field is the ratio $\frac{MSE_{algorithm}}{MSE_{Reg(V)}}$. Lower ratio implies lower MSE. Best ratios are in bold. The results for the regression datasets are averaged over 10 different permutations. The standard deviation is significantly lower in OMKR versions.

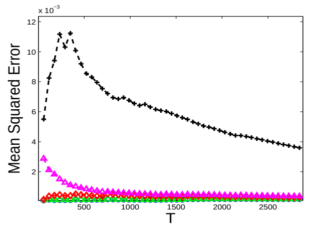
	Widrow Hoff					Norma				
ID	**Reg(V)**	**Reg(H)**	**Uniform**	**Hedge**	**OGD**	**Reg(V)**	**Reg(H)**	**Uniform**	**Hedge**	**OGD**
Regression Datasets										
D1	1.00	**0.85**	1.06	0.89	0.98	1.00	0.36	0.29	0.31	**0.26**
D2	1.00	**0.39**	0.79	**0.39**	0.45	1.00	0.40	0.49	0.40	**0.39**
D3	1.00	**0.69**	3.90	**0.69**	0.82	1.00	0.87	1.55	**0.52**	0.60
D4	1.00	**0.79**	1.13	**0.79**	0.85	1.00	0.77	0.93	0.74	**0.67**
D5	1.00	**0.53**	2.85	0.56	0.62	1.00	**0.21**	0.53	**0.21**	0.22
Time Series Datasets										
D6	1.00	**0.13**	0.50	**0.14**	**0.14**	1.00	0.96	1.68	0.70	**0.57**
D7	1.00	0.96	1.61	0.93	**0.37**	1.00	0.98	0.69	0.23	**0.15**
D8	1.00	**0.96**	12.40	1.65	1.67	1.00	0.73	0.30	**0.15**	0.18
D9	1.00	**0.01**	0.07	**0.01**	**0.01**	1.00	0.79	0.65	**0.18**	**0.17**
D10	1.00	0.83	2.90	0.84	**0.66**	1.00	0.67	1.60	**0.45**	0.54



(a) D1

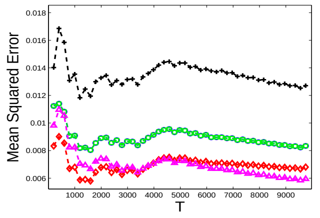


(b) D3

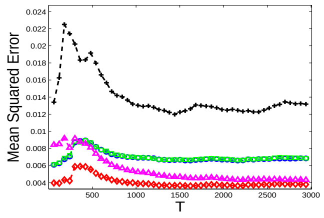


(c) D8

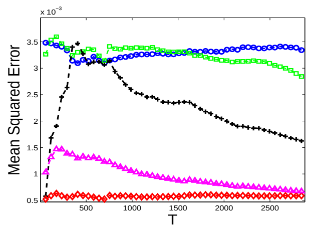
Figure 1: Cumulative Mean Squared Error with time (when Widrow-Hoff is used for regression): All results are displayed for data after the validation stage during which the parameters were determined.



(a) D1

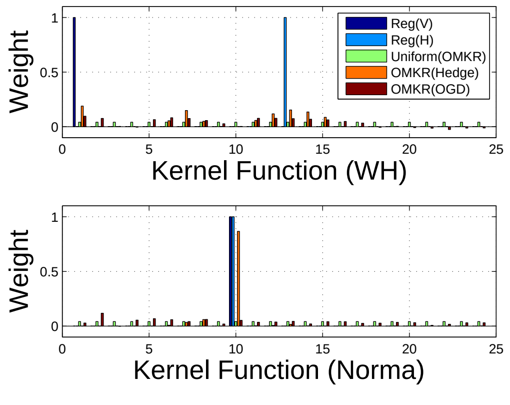


(b) D3

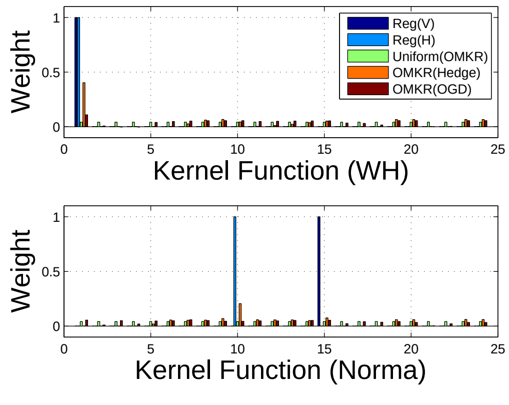


(c) D8

Figure 2: Cumulative Mean Squared Error with time (when Norma is used for regression): All results are displayed for data after the validation stage during which the parameters were determined.



(a) D1



(b) D3

Figure 3: Weight distribution attained by all algorithms

Table 3: OMKR (Hedge) vs Stochastic and Budget Strategies:
Here again, the results of regression datasets are averaged over 10 random permutations. All times are in seconds.

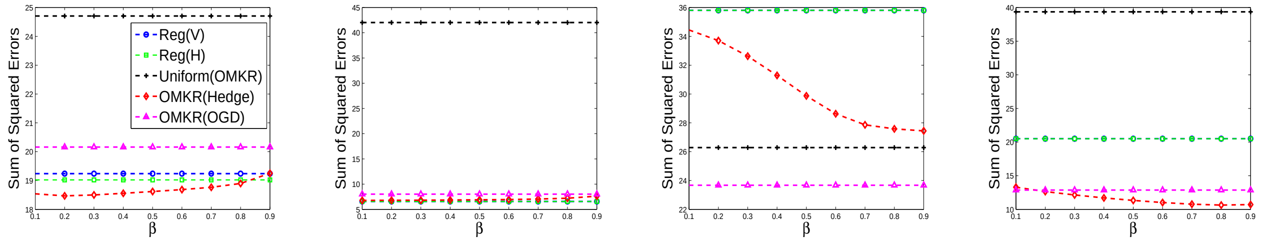
	Widrow Hoff						Norma					
ID	**Determinisitc**		**Stochastic**		**Budget**		**Det OMKR**		**Stochastic**		**Budget**	
	MSE	**Time**	**MSE**	**Time**	**MSE**	**Time**	**MSE**	**Time**	**MSE**	**Time**	**MSE**	**TIME**
	Regression Datasets											
D1	**0.0075**	348	0.0079	39	0.0096	74	0.0121	220	**0.0113**	45	0.0122	68
D2	**0.0209**	707	0.0488	32	0.0436	109	**0.0451**	537	0.0544	45	0.0467	102
D3	**0.0028**	193	0.0035	22	0.0058	54	**0.0049**	159	0.0050	37	**0.0049**	53
D4	**0.0245**	8496	**0.0243**	376	0.0354	385	0.0477	6397	**0.0416**	354	**0.0413**	388
D5	**0.0054**	1950	0.0096	64	0.0122	183	**0.0108**	1338	0.0151	112	0.0116	171
	Time Series Datasets											
D6	**0.0022**	4062	0.0034	146	0.0066	427	0.0058	2611	**0.0034**	182	0.0059	413
D7	**0.0025**	5728	0.0030	831	0.0088	312	**0.0008**	5101	0.0017	1118	**0.0008**	322
D8	0.0003	346	**0.0002**	15	0.0007	117	0.0005	274	**0.0002**	136	0.0005	111
D9	**0.0009**	352	0.0010	56	0.0011	118	**0.0006**	280	0.0010	119	**0.0006**	114
D10	**0.0074**	16	0.0086	3	0.0089	15	**0.0047**	12	0.0086	6	**0.0047**	12

Table 4: OMKR (OGD) vs Stochastic and Budget Strategies:
Here again, the results of regression datasets are averaged over 10 random permutations. All times are in seconds.

	Widrow Hoff						Norma					
ID	**Determinisitc**		**Stochastic**		**Budget**		**Det OMKR**		**Stochastic**		**Budget**	
	MSE	**Time**	**MSE**	**Time**	**MSE**	**Time**	**MSE**	**Time**	**MSE**	**Time**	**MSE**	**TIME**
	Regression Datasets											
D1	**0.0082**	348	0.0086	43	0.0097	76	0.0105	220	**0.0095**	48	0.0105	69
D2	**0.0230**	707	0.0488	35	0.0432	111	**0.0442**	537	0.0521	49	0.0466	105
D3	**0.0034**	193	0.0045	24	0.0043	55	0.0055	159	**0.0044**	40	0.0056	54
D4	**0.0261**	8496	**0.0260**	414	0.0311	393	**0.0006**	6397	0.0322	382	0.0363	396
D5	**0.0060**	1950	0.0109	70	0.0108	187	**0.0115**	1338	**0.0115**	121	0.0121	174
	Time Series Datasets											
D6	**0.0021**	4062	0.0039	160	0.0055	427	**0.0048**	2611	**0.0048**	200	**0.0048**	413
D7	**0.0010**	5728	0.0011	914	0.0023	318	**0.0005**	5101	0.0008	1230	0.0006	328
D8	0.0003	346	**0.0002**	17	0.0004	117	0.0006	274	**0.0002**	149	0.0006	111
D9	**0.0004**	352	0.0005	62	0.0005	118	0.0006	280	**0.0004**	130	0.0006	114
D10	**0.0058**	16	0.0113	3	0.0066	15	**0.0056**	12	0.0063	6	**0.0056**	12

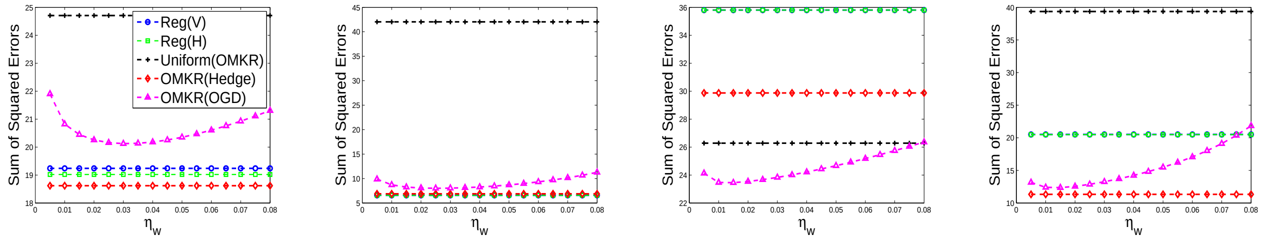
Table 5: Budget OMKR on large time series datasets:
The comparison is against the budget version of all algorithms. All values are MSE achieved by each algorithm. Runtime of D11 was 4000 seconds, and of D12 was 12000 seconds.

	Widrow Hoff					Norma				
ID	**Reg(V)**	**Reg(H)**	**Uniform**	**Hedge**	**OGD**	**Reg(V)**	**Reg(H)**	**Uniform**	**Hedge**	**OGD**
	Large Time Series Datasets									
D11	0.074807	0.015779	0.040779	0.015709	**0.007766**	0.017452	0.012429	0.028378	0.011465	**0.006457**
D12	0.000327	**0.000026**	0.001580	**0.000027**	0.000028	0.000915	0.000637	0.020063	0.000605	**0.000031**



(a) D1 (WH) (b) D3 (WH) (c) D1 (Norma) (d) D3 (Norma)

Figure 4: Sensitivity of OMKR(Hedge) to discount rate parameterβ:
We vary β keeping the performance of all other algorithms fixed.



(a) D1 (WH) (b) D3 (WH) (c) D1 (Norma) (d) D3 (Norma)

Figure 5: Sensitivity of OMKR(OGD) to learning rate η_w:
We vary η_w keeping the performance of all other algorithms fixed.

5. CONCLUSION

This paper proposes a family of OMKR algorithms for kernel based regression using a pool of predefined kernels. They overcome the challenges of existing work which are largely designed for a batch setting and assume that the appropriate kernel function is known. OMKR sequentially learns the kernel based regressor in an online and scalable fashion, and dynamically explores a pool of multiple diverse kernels to avoid problems of poor kernel choice by manual or heuristic selection. Further, OMKR is particularly useful in a time series setting, where the appropriate window size p of an $AR(p)$ process is not known. Varying values of p, each with its own set of predefined kernels, are plugged into the OMKR framework, thus exploring and identifying the best kernel based regressor dynamically. We evaluate the performance of OMKR based on two types of loss functions. Any kernel based regression task can be plugged into the generic OMKR framework, and is likely to achieve a better result than any single kernel. We also propose stochastic and budget techniques to enhance efficiency. Our empirical evaluations show the excellent performance of OMKR, often beating the best prediction function determined in hindsight.

6. REFERENCES

[1] P. Auer, N. Cesa-Bianchi, Y. Freund, and R. E. Schapire. The nonstochastic multiarmed bandit problem. *SIAM Journal on Computing*, 32(1):48–77, 2002.

[2] O. Bousquet and D. J. Herrmann. On the complexity of learning the kernel matrix. *Advances in NIPS*, pages 415–422, 2003.

[3] D. Brugger, W. Rosenstiel, and M. Bogdan. Online svr training by solving the primal optimization problem. *Journal of Signal Processing Systems*, 65(3):391–402, 2011.

[4] G. Cavallanti, N. Cesa-Bianchi, and C. Gentile. Tracking the best hyperplane with a simple budget perceptron. *Machine Learning*, 69:143–167, 2007.

[5] N. Cesa-Bianchi and G. Lugosi. *Prediction, learning, and games*. Cambridge University Press, 2006.

[6] K. Crammer, O. Dekel, J. Keshet, S. Shalev-Shwartz, and Y. Singer. Online passive-aggressive algorithms. *JMLR*, 7:551–585, Dec. 2006.

[7] K. Crammer, J. Kandola, and Y. Singer. Online classification on a budget. In *Advances in NIPS 16*. MIT Press, Cambridge, MA, 2004.

[8] O. Dekel, S. Shalev-Shwartz, and Y. Singer. The forgetron: A kernel-based perceptron on a budget. *SIAM Journal on Computing*, 37(5):1342–1372, 2008.

[9] Y. Freund and R. E. Schapire. A desicion-theoretic generalization of on-line learning and an application to boosting. In *Computational Learning Theory*, volume 904, pages 23–37. Springer Berlin Heidelberg, 1995.

[10] M. Gönen and E. Alpaydın. Multiple kernel learning algorithms. *The Journal of Machine Learning Research*, 12:2211–2268, 2011.

[11] S. C. Hoi, R. Jin, and M. R. Lyu. Learning nonparametric kernel matrices from pairwise constraints. In *ICML*, pages 361–368. ACM, 2007.

[12] S. C. Hoi, R. Jin, P. Zhao, and T. Yang. Online multiple kernel classification. *Machine Learning*, 90(2):289–316, 2013.

[13] S. C. Hoi, M. R. Lyu, and E. Y. Chang. Learning the unified kernel machines for classification. In *KDD*, pages 187–196. ACM, 2006.

[14] S. C. Hoi, J. Wang, and P. Zhao. *LIBOL: A Library for Online Learning Algorithms*. Nanyang Technological University, 2012.

[15] R. Jin, S. Hoi, and T. Yang. Online multiple kernel learning: Algorithms and mistake bounds. In *Algorithmic Learning Theory*, volume 6331, pages 390–404. Springer Berlin Heidelberg, 2010.

[16] H. Kashima, K. Tsuda, and A. Inokuchi. Marginalized kernels between labeled graphs. In *ICML*, volume 3, pages 321–328, 2003.

[17] J. Kivinen, A. Smola, and R. Williamson. Online learning with kernels. *Signal Processing, IEEE Transactions on*, 52(8):2165–2176, 2004.

[18] J. T. Kwok and I. W. Tsang. Learning with idealized kernels. In *ICML*, pages 400–407, 2003.

[19] G. R. G. Lanckriet, N. Cristianini, P. Bartlett, L. E. Ghaoui, and M. I. Jordan. Learning the kernel matrix with semidefinite programming. *J. Mach. Learn. Res.*, 5:27–72, Dec. 2004.

[20] N. Littlestone and M. K. Warmuth. The weighted majority algorithm. In *Foundations of Computer Science, 1989., 30th Annual Symposium on*, pages 256–261. IEEE, 1989.

[21] A. F. Martins, M. A. Figueiredo, P. M. Aguiar, N. A. Smith, and E. P. Xing. Online multiple kernel learning for structured prediction. *arXiv preprint arXiv:1010.2770*, 2010.

[22] E. Moroshko and K. Crammer. A last-step regression algorithm for non-stationary online learning. *CoRR*, abs/1303.3754, 2013.

[23] F. Orabona, J. Keshet, and B. Caputo. The projectron: a bounded kernel-based perceptron. In *ICML*, pages 720–727. ACM, 2008.

[24] N. Sapankevych and R. Sankar. Time series prediction using support vector machines: a survey. *Computational Intelligence Magazine, IEEE*, 4(2):24–38, 2009.

[25] B. Schölkopf and A. J. Smola. *Learning with kernels.* ŞTheŤ MIT Press, 2002.

[26] L. C. S. V. D. Schuurmans and S. W. Caelli. Implicit online learning with kernels. In *NIPS*, volume 19, page 249. MIT Press, 2007.

[27] S. Shalev-Shwartz. Online learning: Theory, algorithms, and applications. 2007.

[28] J. Shawe-Taylor and N. Cristianini. *Kernel methods for pattern analysis*. Cambridge university press, 2004.

[29] A. J. Smola and B. Schölkopf. A tutorial on support vector regression. *Statistics and computing*, 14(3):199–222, 2004.

[30] S. Sonnenburg, G. Rätsch, C. Schäfer, and B. Schölkopf. Large scale multiple kernel learning. *JMLR*, 7:1531–1565, 2006.

[31] U. Thissen, R. Van Brakel, A. De Weijer, W. Melssen, and L. Buydens. Using support vector machines for time series prediction. *Chemometrics and intelligent laboratory systems*, 69(1):35–49, 2003.

[32] V. Vovk. A game of prediction with expert advice. In *COLT*, pages 51–60. ACM, 1995.

[33] B. WIDROW, M. E. HOFF, et al. Adaptive switching circuits. 1960.

[34] Z. Xu, R. Jin, I. King, and M. R. Lyu. An extended level method for efficient multiple kernel learning. In *NIPS*, pages 1825–1832, 2008.

[35] H. Yang, Z. Xu, I. King, and M. R. Lyu. Online learning for group lasso. In *Proceedings of the 27th ICML*, pages 1191–1198, 2010.

[36] P. Zhao, J. Wang, P. Wu, R. Jin, and S. C. Hoi. Fast bounded online gradient descent algorithms for scalable kernel-based online learning. 2012.

[37] J. Zhuang, I. W. Tsang, and S. C. Hoi. A family of simple non-parametric kernel learning algorithms. *JMLR*, 12:1313–1347, 2011.

[38] M. Zinkevich. Online convex programming and generalized infinitesimal gradient ascent. 2003.