Temporal Kernel Descriptors for Learning with Time-sensitive Patterns

Doyen Sahoo[*] Abhishek Sharma[†] Steven C.H. Hoi[‡] Peilin Zhao[§]

Abstract

Detecting temporal patterns is one of the most prevalent challenges while mining data. Often, timestamps or information about when certain instances or events occurred can provide us with critical information to recognize temporal patterns. Unfortunately, most existing techniques are not able to fully extract useful temporal information based on the time (especially at different resolutions of time). They miss out on 3 crucial factors: (i) they do not distinguish between timestamp features (which have cyclical or periodic properties) and ordinary features; (ii) they are not able to detect patterns exhibited at different resolutions of time (e.g. different patterns at the annual level, and at the monthly level); and (iii) they are not able to relate different features (e.g. multi-modal features) of instances with different temporal properties (e.g. while predicting stock prices, stock fundamentals may have annual patterns, and at the same time factors like peer stock prices and global markets may exhibit daily patterns). To solve these issues, we offer a novel multiple-kernel learning view and develop *Temporal Kernel Descriptors* which utilize Kernel functions to comprehensively detect temporal patterns by deriving relationship of instances with the time features. We automatically learn the optimal kernel function, and hence the optimal temporal similarity between two instances. We formulate the optimization as a Multiple Kernel Learning (MKL) problem. We empirically evaluate its performance by solving the optimization using Online MKL.

1 Introduction

Mining temporal patterns is one of the most prevalent problems in data mining, and has immense applications and practical utility. Detecting how events are associated or linked with timestamp features can greatly improve our predictive ability and decision making. For example, estimating the weather conditions can help in agriculture, event planning etc.; estimating trends in demands for resources can help in resource allocation (e.g. what time of the day would we have high demand for car rentals, what would be the demand for resources in a super market at different times); estimating stock prices can help us make better investment decisions. It is therefore critical to have techniques that can comprehensively detect temporal patterns from the data.

[*]Singapore Management University, doyensahoo.2014@smu.edu.sg
[†]Singapore Management University, abhisheksh.2014@smu.edu.sg
[‡]Singapore Management University, chhoi@smu.edu.sg
[§]Institute for Infocomm Research, A*STAR, zhaop@i2r.a-star.edu.sg

Several existing machine learning techniques consider timestamp features as ordinary features, and directly plug them into the learning method. Some alternate techniques explicitly address the temporal nature of data (without timestamp usage), by weighting (e.g. exponential weighting) the instances based on recency in the optimization formulation. These approaches suffer from three major drawbacks: (i) they ignore the cyclical or periodic nature of timestamp features (e.g. hour feature repeats itself after every 24 iterations); (ii) they can not recognize temporal patterns at different resolutions of time (e.g. distinction is required between patterns observed at different days of the week, and different hours of the day); and (iii) they do not associate different multi-modal features with different timestamp resolutions (e.g. for stock price prediction, the fundamentals may exhibit monthly patterns, whereas global markets and peer stocks may exhibit daily patterns). To solve these problems, in this paper we develop novel multi-resolution kernel functions called *Temporal Kernel Descriptors* which use kernels to describe the temporal patterns in the data by appropriately linking events to timestamps at various resolutions of time. We automatically learn the optimal kernel function that can appropriately measure the temporal similarity between instances by formulating the problem as a Multiple Kernel Learning (MKL) optimization [3, 19, 33].

We motivate this problem further through a real world example of bike demand prediction at a given time [8] (See Figure 1). The demand may be affected by the hour of the day, or day of the week, or be a function of both in conjunction. Likewise, (multimodal) factors like weather, temperature, or if the day is a holiday (these can be considered heterogeneous data sources) will also be responsible in varying the bike demand. Additionally, these multi-modal factors will have a different effect based on the timestamp (at different resolutions). We aim to design *Temporal Kernel Descriptors* that can not only adapt according to periodic properties of timestamp features at different resolutions, but also be able to find appropriate association of different modalities (or heterogeneous features) with each of the timestamp resolutions. First, we introduce *temporal kernels* which address the problem of time features having a periodic nature. We then extend this to *multi-resolution temporal kernels* to measure time similarity at different resolutions. Finally, we design *temporal kernel descriptors* that associate various modalities to different resolutions of time. We al-

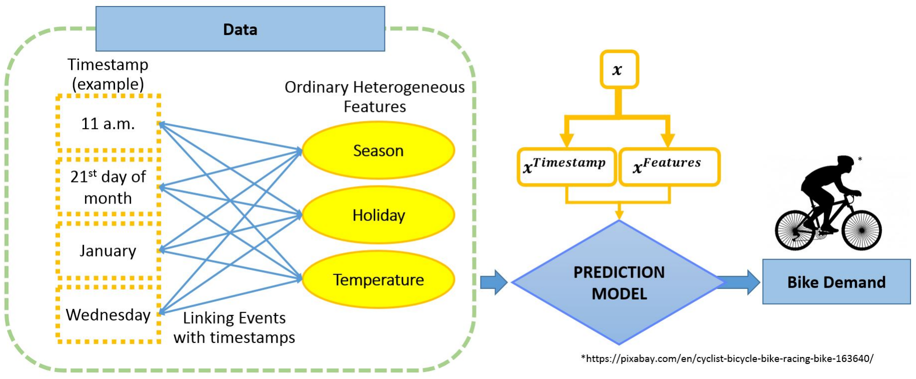


Figure 1: To estimate the bike demand, we aim to learn the association between each multi-modal (heterogeneous) feature set with each resolution of time. The prediction model then accepts an input and estimates the demand.

so devise a method to use these descriptors in absence of timestamps. We formulate the optimization as a Multiple Kernel Learning problem, in order to automatically learn the optimal temporal similarity between two instances. We also empirically validate the superiority of using these kernel descriptors in detecting temporal similarity as compared to traditional state of the art linear and kernel methods.

2 Related Work

There are two main categories of work related to our paper: Temporal Pattern Mining and Multiple Kernel Learning. Temporal data mining has been studied extensively in literature [2, 20, 25]. We restrict our discussion to kernel methods or timestamp usage. Kernel methods, particularly support vector regression have found success in several time-series applications including finance [34], electricity load forecasting [17], credit rating [15], machine reliability forecasting [37], control systems and signal processing [11] and online learning [29]. Most of these techniques primarily focus directly using the traditional kernels, and applying them directly to the features. Periodic kernel functions have also been used to address issues temporal periodicity in time series [22, 23]. Additionally, some attempts have been made to explicitly address the temporal issues by modifying the objective function like exponential weighting [35], giving more importance to recent data. A similar approach is seen in several online learning methods, in which they adapt to the changing pattern [6]. Another related work is the usage of autoregressive kernels for time series prediction [5]. These methods address a slightly different problem and do not exploit timestamp information to improve predictability. Some efforts have been made to use tree-kernels for linking events to timestamps [24, 14], but here the focus is primarily on natural language processing. None of these methods consider the temporal patterns of multi-modal data at multiple resolutions of time.

The other closely related area of work is Kernel Learning which deals with learning the optimal similarity between instances. Kernel methods are more effective than linear methods when learning a nonlinear target hypothesis [31]. They are also used as notions of similarity. In practice several types of kernel functions exist, with a variety of parameters. It is usually not known before-hand which kernel would be suitable for the task. Further, often data sources are heterogeneous, implying different kernels may be suited to different features (e.g. if an instance has features that may be numerical or strings). In addition, the usage of a single kernel usually restricts the learning capacity by not leveraging on information that could be gotten from more kernels. To tackle these problems, learning the kernel function evolved as a promising research direction where the most prevalent method is Multiple Kernel Learning (MKL) [19] which learns the optimal combination of multiple kernels. Most of these [19, 33, 12] focus on batch learning settings, and are often very computationally expensive. Many techniques have been proposed to enhance the efficiency of MKL including SimpleMKL [28], extended level method [36], and mirror descent [1]. Despite improved efficiency, these methods lose practical utility when confronted with lot of data or a large pool of kernel functions. As a result online learning, a field that was designed for scalable algorithms, which process data sequentially, was extended to multiple kernel learning. These include online multiple kernel learning in [16, 13, 30] where the multiple kernel combination is learnt using the Hedge Algorithm [9, 10] or the online gradient descent approach [38]. [21] studied online MKL for structured prediction. These methods use fast and scalable online learning techniques to solve multiple kernel learning in an online manner, at the cost of not being able to reach a global optimum. Unfortunately, none of these techniques consider the usage of kernels to better describe the temporal nature of the multi-modal data at multi-resolution timestamps.

3 Temporal Kernel Descriptors

3.1 Problem Setting Consider a set of N instances $\mathbf{x}_i$, for $i = 1, \ldots, N$, with each instance labeled with a target value y_i, where $y_i \in \{1, -1\}$ for classification or $y_i \in \mathcal{R}$ for regression tasks. Each instance comprises two sets of features: $\mathbf{x}_i^T$, which is a timestamp, or the time at which the event took place, e.g., $\mathbf{x}_i^T = \{12^{th} Jan, 2015|10 : 28 : 31\}$ (or any other customized time feature designed by the user); and $\mathbf{x}_i^F$ are those standard features in a typical learning setting that describe the instance.

Based on these two sets of features we aim to learn a kernel based prediction model $\mathbf{f}(\mathbf{x}_i)$ which can optimally predict the target variable y_i. A prediction on instance $\mathbf{x}_i$ with target y_i suffers a loss which is denoted by $\ell(\mathbf{f}, (\mathbf{x}_i, y_i))$. Typically this loss function is convex, and can be application dependent. For example, for binary classification, it can be the hinge loss: $\ell(\mathbf{f}, (\mathbf{x}_i, y_i)) = \max(0, 1 - y_i \mathbf{f}(\mathbf{x}_i))$; and for regression it can be the squared loss: $\ell(\mathbf{f}, (\mathbf{x}_i, y_i)) = \frac{1}{2}(\mathbf{f}(x_i) - y_i)^2$.

Our goal is to learn a function that minimizes this loss over all the instances in the dataset. This can be cast into the following (SVM-like) optimization problem:

$$\min_{\mathbf{f}} \frac{1}{2}||\mathbf{f}||^2 + C \sum_{i=1}^{N} \ell(\mathbf{f}, (\mathbf{x}_i, y_i))$$

where the first term is the regularizer penalizing the complexity of the model, and the C is the tradeoff parameter. In the following, we will describe the kernels used to learn this kernel-based prediction function.

3.2 Temporal Kernels Kernels have found success in many algorithms and applications [32] owing to their ability to detect nonlinear patterns. Kernels implicitly map the input space into a high dimensional space (also called the Reproducing Kernel Hilbert Space (RKHS)), and allow the algorithms to detect linear patterns in this new space. This enables the algorithms to infer nonlinear patterns in the original space. What makes kernels efficient and popular is the *kernel trick*, which allows us to skip the explicit high dimensional mapping, and learn the prediction model in the RKHS while operating in the original space. The kernel trick is used in Kernel Functions, which compute the dot product between instances in a high dimensional space. Popularly used kernel functions include: Polynomial Kernels $\kappa(\mathbf{x}_1, \mathbf{x}_2) = (c + \mathbf{x}_1^\top \mathbf{x}_2)^d$ (c and d are parameters), and Gaussian Kernels $\kappa(\mathbf{x}_1, \mathbf{x}_2) = \exp(-\frac{||\mathbf{x}_1 - \mathbf{x}_2||^2}{2\sigma^2})$ (σ is the bandwidth parameter). Since kernel functions essentially compute the dot product between two instances, they are often viewed as similarity measures, i.e., the kernel function on two instances returns the similarity between the two instances, where different types of kernels (with different parameters) denote different notions of similarity.

Using this similarity view of kernels, how should we compute similarity between two timestamps? Unfortunately traditional kernels e.g. linear kernel, or (Euclidean) distance based kernels (e.g. Hat Kernel, Gaussian Kernel, etc.) give poor quality similarity scores for time values owing to the cyclical nature of the many time measures (e.g. hour value repeats itself after 24 iterations, seconds value repeats itself after every 60 iterations). Intuitively we can comprehend that 2300 hours and 0100 hours occur at similar times of the day, but traditional kernels consider these two times as very dissimilar due to the large Euclidean distance. To address this problem, we design a *temporal kernel* which can account for the cyclical or periodic nature of timestamps. Let $\mathbf{x}_1^t$ and $\mathbf{x}_2^t$ be two time stamp instances with a cycle length C_t (for hours $C_t = 24$). The temporal kernel similarity measure based on the Gaussian kernel for the the two instances is given by:

$$\kappa_t(\mathbf{x}_1^t, \mathbf{x}_2^t) = \exp - \frac{(\min(|\mathbf{x}_1^t - \mathbf{x}_2^t|, C_t - |\mathbf{x}_1^t - \mathbf{x}_2^t|))^2}{2\sigma^2}$$

Here σ is the bandwidth parameter as used in the Gaussian Kernel. This new temporal kernel measures the Euclidean distance between two time values from the shorter sides for the cycle, and thus gives a better similarity measure. We note that this is similar, albeit a more simplified version of the Mackay Periodic kernel function. An example of this is graphically illustrated in Figure 2.

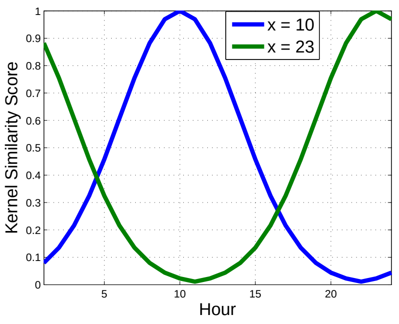


Figure 2: Temporal Kernel: The kernel similarity score for $\mathbf{x} = 10$ and $\mathbf{x} = 23$ for $\sigma = 4$. Here, $\mathbf{x}$ represents hours.

3.3 Multi-resolution Temporal Kernels Having addressed the periodic nature of temporal patterns through temporal kernels, we are still faced with the challenge of addressing time similarity measures when multi-resolution timestamps are provided. For example, what would be the kernel similarity between 09:05 a.m. and 11:35 a.m.? Should the similarity be high considering that the time stamps are only a few hours apart, or should the similarity score be low as the minutes feature of the time stamps are far apart? A naive approach would be to convert everything to a standardized unit of time (convert everything to hours), and try to use the temporal kernels. However, temporal patterns at different resolutions may get ignored. The data may exhibit patterns at different resolutions of time - for example, there is a global pattern in every hour and a local pattern in every minute (within each hour). We want to be able to capture the global pattern (changes in every hour), as well as the local pattern (changes at a minutes level).

To do this, we consider each of the specific time features as heterogeneous (or multi-modal) and define the multi-resolution temporal kernel similarity between two stamps as the positively weighted linear combination of kernel similarity of each timestamp feature. Let $\mathbf{x}_1^T$ and $\mathbf{x}_2^T$ be two timestamp instances comprising $T > 1$ features (i.e. $\mathbf{x}_1^T = [x_1^1, \ldots, x_1^T]$). The *multi-resolution temporal kernel* similarity based on multiple kernels between the instances is given by:

$$\begin{aligned}
&\kappa_{Multi-resolutionTemporalKernel}(\mathbf{x}_1^T, \mathbf{x}_2^T)\\
&\quad = \alpha_1\kappa_1(\mathbf{x}_1^T, \mathbf{x}_2^T) + \cdots + \alpha_T\kappa_T(\mathbf{x}_1^T, \mathbf{x}_2^T)\\
&\quad = \sum_{t=1}^{T} \alpha_t\kappa_t(\mathbf{x}_1^T, \mathbf{x}_2^T)\\
&\quad \text{subject to } \alpha_t \geq 0
\end{aligned}$$

$\alpha_t \geq 0$ is enforced to make sure that the resultant kernel function is positive semi-definite keeping the learnt kernel a valid Mercer Kernel. Each kernel function used in the above equation has its own bandwidth parameter and cycle length (C_t), depending on the application and the timestamp feature being used used from the instance $\mathbf{x}^T$.

3.4 Temporal Kernel Descriptors Using appropriate temporal kernels, we now associate the timestamps to the heterogeneous feature sets. First we define feature level similarity between two instances. Let $\mathbf{x}_1^F$ and $\mathbf{x}_2^F$ be two instances (without timestamp information) comprising $F > 1$ feature sets (i.e. $\mathbf{x}_1^F = [x_1^1, \ldots, x_1^F]$, where every x_1^f represents a multi-modal data source, and it can have one or more features). The similarity between these two instances is given as a positively weighted linear combination of kernel similarity scores of each feature set. It is denoted as $\kappa_{FeatureKernel}$ and is given by:

$$\begin{aligned}
&\kappa_{FeatureKernel}(\mathbf{x}_1^F, \mathbf{x}_2^F)\\
&\quad = \beta_1\kappa_1(\mathbf{x}_1^F, \mathbf{x}_2^F) + \cdots + \beta_F\kappa_F(\mathbf{x}_1^F, \mathbf{x}_2^F)\\
&\quad = \sum_{f=1}^{F} \beta_f\kappa_f(\mathbf{x}_1^F, \mathbf{x}_2^F)\\
&\quad \text{subject to } \beta_f \geq 0
\end{aligned}$$

$\beta_t \geq 0$ is enforced to make sure that the resultant kernel function is positive semi-definite keeping the learnt kernel a valid Mercer Kernel. Each kernel function used can be of any suitable type based on the application and the heterogeneous source. κ_f computes the kernel similarity of instances based on the f^{th} set of heterogeneous features.

Finally, we define *Temporal Kernel Descriptors* (TKD) which depict the instance based similarity based on the time similarity and feature similarity by linking similarity scores of both the timestamp features and multi-modal data features to obtain a multi-resolution kernel similarity function. This similarity is scored as the product of multi-resolution temporal kernel similarity and the instance based feature similarity. It is denoted by $\kappa^{TKD}(\mathbf{x}_1, \mathbf{x}_2)$ for instances $\mathbf{x}_1 = [\mathbf{x}_1^T, \mathbf{x}_1^F]$ and $\mathbf{x}_2 = [\mathbf{x}_2^T, \mathbf{x}_2^F]$ and is given by:

$$\begin{aligned}
(3.1)\ \ &\kappa_{TKD}(\mathbf{x}_1, \mathbf{x}_2)\\
&= \kappa_{HTK}(\mathbf{x}_1^T, \mathbf{x}_2^T) \cdot \kappa_{FK}(\mathbf{x}_1^F, \mathbf{x}_2^F)\\
&= \left(\sum_{t=1}^{T} \alpha_t\kappa_t(\mathbf{x}_1^T, \mathbf{x}_2^T)\right)\left(\sum_{f=1}^{F} \beta_f\kappa_f(\mathbf{x}_1^F, \mathbf{x}_2^F)\right)\\
&= \sum_{t=1}^{T}\sum_{f=1}^{F} \alpha_t\beta_f\left(\kappa_t(\mathbf{x}_1^T, \mathbf{x}_2^T) \cdot \kappa_f(\mathbf{x}_1^F, \mathbf{x}_2^F)\right)\\
&= \sum_{t=1}^{T}\sum_{f=1}^{F} w_{tf}\left(\kappa_t(\mathbf{x}_1^T, \mathbf{x}_2^T) \cdot \kappa_f(\mathbf{x}_1^F, \mathbf{x}_2^F)\right)
\end{aligned}$$

As can be seen, the TKD similarity is a linear combination of a set of different kernel functions. Since $\alpha_t \geq 0$ and $\beta_f \geq 0$, $w_{tf} = \alpha_t \cdot \beta_f \geq 0$. The final kernel function is a conic combination of a product of kernel functions. Both kernel product and conic combination preserve a positive semi-definite kernel, which means the final kernel function is also positive semi-definite, and is a Mercer Kernel.

Each product of two kernel functions establishes a link or an association between a timestamp feature and each of the multi-modal data sources. This helps us associate events for a specific set of multi-modal features with the time at which they occurred and allows us to capture the temporal similarity in the data at various resolutions of time for each modality. Our next step is to learn the optimal coefficients w_{tf} for each of the multiple kernel functions, in order to derive the optimal kernel function.

3.5 Optimization and Algorithms We wish to minimize human intervention in determining the best kernel function, and we want an optimization objective that can automatically find this best kernel function. Our objective is to learn the optimal coefficients w_{tf} such that the optimal kernel function can minimize the loss over the entire data. Given T timestamp features, and F instance features, we get a total of $m = T \times F$ kernel functions, whose weighted combination needs to be learnt. Following a structural risk minimization principle, we can cast this problem into the following multiple kernel learning (MKL) optimization:

$$(3.2) \min_{\mathbf{w}\in\Delta} \min_{\mathbf{f}\in\mathcal{H}_\kappa} \frac{1}{2}||\mathbf{f}||^2_{\mathcal{H}_\kappa(\mathbf{w})} + C\sum_{i=1}^{N} \ell(\mathbf{f},(\mathbf{x}_i,y_i))$$

where $\Delta = \{\mathbf{w} \in \mathcal{R}^m_+ | \sum_{j=1}^m w_j = 1\}$ and $\kappa(w)(\mathbf{x}_1,\mathbf{x}_2) = \sum_{t=1}^T \sum_{f=1}^F w_{tf} \left(\kappa_t(\mathbf{x}_1^T,\mathbf{x}_2^T) \cdot \kappa_f(\mathbf{x}_1^F,\mathbf{x}_2^F)\right)$. Further $\ell(\mathbf{f},(\mathbf{x}_i,y_i))$ is a convex loss function that can suitably chosen based on the task at hand. There are many techniques to solve this optimization involving structural risk minimization with multiple kernels [33, 27, 36, 12].

Unfortunately, MKL optimization shown above is computationally very expensive. In particular when data has a lot of instances, or the number of kernel functions used is large, the training cost (and re-training cost) is very high. In fact, in the scenario of multi-modal data with multi-resolutions of timestamps, the number of kernels quickly expands as a product of the number of modalities and number of timestamp resolutions. To alleviate this problem, Online MKL [16, 13, 30] has evolved as a promising research direction, in which both the kernel predictions and their optimal combination are learnt in an online or sequential fashion. In the following we briefly discuss this approach in order to learn a kernel-based prediction function for time-sensitive patterns using Temporal Kernel Descriptors.

The idea of OMKL is to first define a pool of base kernels. For our task, the predefined pool of base kernels are the $m = T \times F$ kernels. The entire learning task is done in the online setting. This means that the instances arrive sequentially. In every iteration, an instance $\mathbf{x}_i$ is revealed to the model. The model makes a prediction $\hat{y_i} = \mathbf{F}_i(\mathbf{x}_i)$ in each iteration. Subsequently, the environment reveals the true value y_i. As a result, the model suffers loss $\ell(\mathbf{F}_i,(\mathbf{x}_i,y_i))$. Finally, the model updates itself based on this loss, with the objective to achieve the lowest possible loss across all the instances. The main task is to decide how to do the update in such a manner that we are not only learning the optimal prediction function, but also the optimal combination of multiple kernels simultaneously. OMKL approaches this problem with a two-step procedure in every iteration: (i) First each kernel predictor is updated based on the loss it has individual suffered; and (ii) Second, the combination weights of the multiple kernel predictors is updated based on loss suffered by not having the optimal combination.

Learning the Kernel Predictor: An online kernel model with the prediction function $\mathbf{f}(\mathbf{x})$ is updated by gradient descent [38, 18] with learning rate parameter η when the model suffers loss. Such a prediction model is learnt for *each* of the $m = T \times F$ kernels in the predefined pool. The loss for classification can be the hinge loss ($\ell(\mathbf{f},(\mathbf{x},y)) = \max(0, 1 - y(\mathbf{f}\cdot\mathbf{x}))$, and for regression, we consider the squared loss ($\ell(\mathbf{f},(\mathbf{x},y)) = (\mathbf{f}\cdot\mathbf{x} - y)^2$). The update is made as follows:

$$\begin{aligned}\mathbf{f}_{i+1}(\mathbf{x}) &= \mathbf{f}_i(\mathbf{x}) - \eta\nabla_{\mathbf{f}}\ell(\mathbf{f}_i,(\mathbf{x}_i,y_i)) \\ &= \mathbf{f}_i(\mathbf{x}) + \eta y_i \kappa^{TKD}(\mathbf{x}_i,\mathbf{x})\text{(for classification)} \\ &\quad \text{OR} \\ &= \mathbf{f}_i(\mathbf{x}) - \eta y_i(\mathbf{f}_i(\mathbf{x}_i) - y_i)\kappa^{TKD}(\mathbf{x}_i,\mathbf{x})\text{(for regression)}\end{aligned}$$

At the end of each online learning round, we can express the prediction function as a kernel expansion [31]:

$$(3.3) \qquad \mathbf{f}_{i+1}(\mathbf{x}) = \Sigma_{j=1}^i \lambda_j \kappa^{TKD}(\mathbf{x}_j,\mathbf{x})$$

where the λ_i coefficients are computed based on the update rule. If a non-zero loss was suffered on the i^{th} instance, then $\lambda_i \neq 0$ and the instance becomes a support vector; and if no loss is suffered, the i^{th} instance is not a support vector for which $\lambda_i = 0$. As can be seen, the final prediction of a single kernel is linear combination of kernel similarity of the instance to be predicted with all existing support vectors.

Learning the Multiple Kernel Combination: Having designed the methodology to optimize each kernel predictor, all of them should be suitably combined to give the final prediction. The final prediction is a weighted combination of the $m = T \times F$ kernel predictors, given by:

$$(3.4) \qquad \hat{y_i} = \mathbf{F}_i(\mathbf{x}_i) = \frac{\sum_{k=1}^m w_i^k(\mathbf{f}_i^k(\mathbf{x}_i))}{\sum_{k=1}^m w_i^k}$$

To update combination of weights $\mathbf{w} = (w^1,\ldots,w^m)^\top$, where w^i is set to $1/m$ at the beginning of the learning task, we use the *Hedge* algorithm [10, 13]. At the end of each learning iteration, the weights are updated by:

$$(3.5) \qquad w_{i+1}^k = w^k \beta^{\ell(\mathbf{f}_i^k;(\mathbf{x}_i,y_i))}$$

where $\beta \in (0,1)$ is the discount rate Hedge parameter, and $\ell(\mathbf{f}_i^k;(\mathbf{x}_i,y_i)) \in (0,1)$ represents the loss suffered by the kernel predictor. At the time of prediction, the weights are normalized to a distribution, ensuring that the weights of the prediction sum up to 1. Similarly, other linear online learning methods could be adopted to learn the optimal combination.

3.6 Analysis In this part, we will briefly discuss some properties of OMKL with Temporal Kernel Descriptors.

Theoretical Analysis: We derive a loss bound for OMKL with Temporal Kernel Descriptors. We assume $\kappa(\mathbf{x},\mathbf{x}) \leq 1$ for all κ and $\mathbf{x}$. We define the optimal objective value for the kernel $\kappa_k(\cdot,\cdot)$ denoted by $O(\kappa_k,\ell,\mathcal{D})$ with respect to the dataset $\mathcal{D}$ as the regret of an online learning algorithm used for a single kernel predictive model. Online gradient descent gives us the regret of the algorithm with respect to the best linear predictor in the Reproducible Kernel Hilbert space induced by kernel κ_k. Since Online Gradient Descent is a no regret algorithm, the regret tends to zero, as the number of instances N goes to infinity.

LEMMA 1. *On processing a sequence of N instances, the cumulative loss suffered by the OMKL using temporal kernel descriptors, where each kernel predictor is updated by online gradient descent is bounded as*

$$L_{TKD} \leq \frac{\ln(\frac{1}{\beta})}{1-\beta} \min_{1\leq k\leq m} O(\kappa_k, \ell, \mathcal{D}) + \frac{\ln m}{1-\beta}$$

L_{TKD} denotes the cumulative loss suffered by the algorithm. By setting $\beta = \frac{\sqrt{N}}{\sqrt{N}+\sqrt{\ln m}}$, we get:

$$L_{TKD} \leq \left(1 + \sqrt{\frac{\ln m}{N}} \min_{1\leq k\leq m} O(\kappa_k, \ell, \mathcal{D}) + \ln m + \sqrt{N \ln m}\right)$$

Proof. The proof combines the Zinkevich theorem [38] and the bound for Hedge Algorithm [10]. The Zinkevich algorithm gives us a no regret algorithm for a particular Reproducible Kernel Hilbert Space induced by a particular kernel. Using a set of m kernels, and updating their weight distribution via the Hedge algorithm, we can directly plug the regret into the loss bound of Hedge algorithm as shown in Equation (3.6). Then, optimally choosing a value for discount rate parameter β we get the result in the lemma. This tells us that OMKL using temporal kernel descriptors in the worst case scenario will converge in performance to the most informative (or the best) kernel descriptor (though in practice, using complimentary information from different kernels is likely to enhance the performance).

Time Complexity: In the worst case scenario, when all instances become support vectors for a single kernel, the computational complexity of each iteration for each kernel is in $O(N)$. Repeating this for N iterations the time complexity is in $O(N * N) = O(N^2)$. Further, these operations have to be performed for all $m = T \times F$ kernels, which means the time complexity of running OMKL is in $O(mN^2)$. An unbounded number of support vectors can make the algorithm less practical, despite a polynomial running time. Several techniques in literature have been proposed to address this issue based on budget approximations [4, 7, 26]. These techniques require a prespecified budget B, which is the maximum number of support vectors a kernel prediction model can store. This budget reduces time complexity of a single kernel predictor on the entire data from $O(N^2)$ to $O(NB)$, and consequently reduces time complexity of OMKL to $O(mNB)$. This is linear in the number of instances N, and hence scales well with large number of instances (as compared to batch MKL methods).

Usage in absence of timestamps: The key to detecting temporal patterns through temporal kernel descriptors lies in having suitable temporal kernels. The temporal kernel descriptors that we have presented in this paper are primarily designed for settings where timestamps are available. However, many applications, despite having a temporal nature, may not have timestamps. A new challenge surfaces - how to use temporal kernels in such scenarios? To address this issue we propose user-defined timestamps, which are parameterized by cycle length C_t. We assume that the data instances arrive sequentially, and in a chronological order (i.e. the instances that occurred first, arrive first). The user can specify temporal cycles as:

$$\mathbf{C}_T = (C_1, C_2, \ldots, C_T)^\top$$

Using these cycle length values, we can design appropriate temporal kernel descriptors. The parameters $\mathbf{C}_T$ can be determined through some simple validation technique to determine appropriate cycle lengths.

4 Experiments

4.1 Experimental Setting

4.1.1 Baselines We perform our analysis for regression tasks, by choosing a squared loss function. In principle, this can be trivially extended to classification tasks as well. We compare the performance of traditional kernels against the proposed Temporal Kernel Descriptors.

Traditional Kernels: First set is the baselines, which comprise traditional usage of kernel methods. These include simple online **Linear** (linear kernel) regression based on gradient descent [38]. We also perform online regression using a variety of kernel functions. We define diverse set of 10 kernels which include three polynomial kernels $\kappa(x, y) = (x^T y)^p$ of degree parameter $p = 1, 2, 3, 4$, five RBF kernels ($\kappa(x, y) = e^{(\frac{-||x-y||^2}{2\sigma^2})}$) of kernel width parameter $\sigma = 2^{-2}, 2^{-1}, 2^0, 2^1, 2^2$, and a sigmoid kernel($\kappa(x, y) = \tanh(xy)$). The algorithm **Best Kernel(Val)** denotes the performance of single kernel online regression, where the choice of kernel has been determined by validating all kernels on first 10% of the instances. **Best Kernel** follows the same principle, but denotes the best kernel choice determined in hindsight (hence usually not realistically attainable). The final algorithm in this set is Online Multiple Kernel Regression (**OMKR**) [30], which tries to learn the optimal combination of multiple kernels (all 10 kernels in this case) to make the optimal prediction. **Temporal Kernel Descriptors**: The next set of 3 algorithms make use of temporal

Table 2: Final Mean Squared Error of algorithms on datasets with time-stampsand on time-series datasets. The first 4 are based on traditional kernels. The last 3 algorithms are based on Temporal Kernels. The best performance is in bold.

	Applications with Time-stamps		**Time Series Datasets**	
Algorithms	**Bike Demand**	**Twitter Traffic**	**Astrophysical**	**Synthetic**
Linear	0.1925	0.2591	0.0088	0.0111
Best Kernel (Val)	0.0379	0.0242	0.0088	0.0111
Best Kernel	0.0379	0.0242	0.0088	0.0091
OMKR	0.0380	0.0244	0.0080	0.0090
(Temporal+Feature)Kernels	0.0214	0.0048	0.0077	**0.0081**
TKD(Uniform)	0.0420	0.0089	0.0101	0.0176
TKD(Hedge)	**0.0209**	**0.0046**	**0.0071**	0.0085

kernel descriptors. **(Temporal+Feature) Kernels** essentially uses multi-resolution temporal kernels and regular feature kernels as multi-modal kernel functions (and does not multiply or link the temporal properties to each multi-modal feature like in Equation (3.1)). This algorithm aims to see the direct advantage of using multi-resolution temporal kernels over existing methods. Like in our proposed method, these kernels are also combined using the Hedge algorithm. The next two algorithms are based on temporal kernel descriptors, i.e., to evaluate the advantage of associating different multi-modalities with different time features, and finding temporal patterns among those (by multiplying temporal kernels with feature kernels like in Equation (3.1)). The first is **TKD(Uniform)** where all the kernel descriptors are equally weighted, and the second algorithm is **TKD(Hedge)** (our proposed method), where the weights of the kernel descriptors are learnt using Hedge Algorithm (as outlined in the previous section). For all gradient descent algorithms, we set the learning rate $\eta = 0.1$, and for Hedge, we set the discount rate parameter $\beta = 0.5$. We also set a budget for online kernel methods $B = 1000$ support vectors. These parameters are common to all algorithms, and hence changes in them would not significantly affect our results of comparison between temporal kernels and traditional kernel methods.

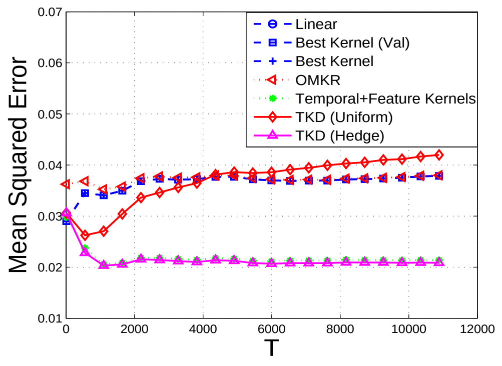
4.1.2 Data We perform our experiments on 2 datasets, that are applications in which time-stamp data is available. First, is the **Bike Demand** dataset obtained from UCI repository. The goal is the predict the bike demand based on factors such as weather, humidity, whether a holiday, etc. We use the Day of the Week (Cycle = 7), and Hour of the day (Cycle = 24), as the multi-resolution temporal kernels. Second, is the **Twitter Traffic** dataset, which we have collected ourselves. The aim is to predict the number of tweets in a specific hour. The entire dataset was processed after analyzing over 68 million tweets from June, 2012 to May, 2013. All tweets are from micrologies who identified themselves as software developers. Again Day of the Week and Hour of the day were used for computing temporal kernels. We also evaluated our method on univariate time series data obtained from Santa Fe Time Series Competition. Here, timestamps are absent, and we arbitrarily designed our own temporal kernels for the task. The two datasets are **Astrophysical** Data, and a **Synthetic** dataset. For both of them, the past 20 values were considered as the input features to predict the next value. In our experiments, we scaled all features (including the target) to lie in $[0, 1]$. For all our datasets, we use 2 resolutions of timestamps, and 2 multi-modal data sources based on the type of features available. We also use an additional modality of features to obtain a trivial kernel function that always evaluate to 1. When this is multiplied by a multi-resolution temporal kernel, it gives us the ability to use pure temporal kernels in our final model as well. The parameter cycle length C_T, and the bandwidth parameter σ for each kernel was set via a grid search on validation data. The other relevant details of the datasets can be seen in Table 1 (here the instance features are combined number of features from multi-modal sources).

Table 1: Datasets used in experiments

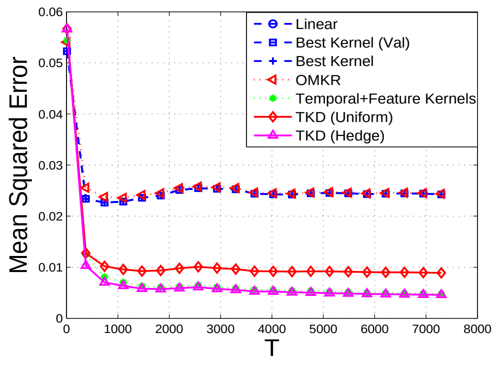
Dataset	Instances	Features	Timestamp Features
Bike	10884	13	Hour, Day of Week
Twitter	7296	12	Hour, Day of Week
Astrophysical	598	20	User Defined
Synthetic	50000	20	User Defined

4.2 Results and Discussion The final mean squared error achieved by the algorithms on the datasets are shown in Table 2, and the performance of the algorithms as the number of instance increases in an online learning setting can be visualized in Figure 3. It can be seen that the algorithms using temporal kernels achieve a significantly lower mean squared error than the algorithms that use traditional kernels. This impact is particularly much more significant in Applications with Time-stamps, where the MSE achieved by our proposed method is almost 50% of traditional kernels for Bike Demand, and for Twitter Traffic Prediction, the MSE is lower than 20% of the best of the traditional kernel methods. This indicates that the patterns are indeed time-

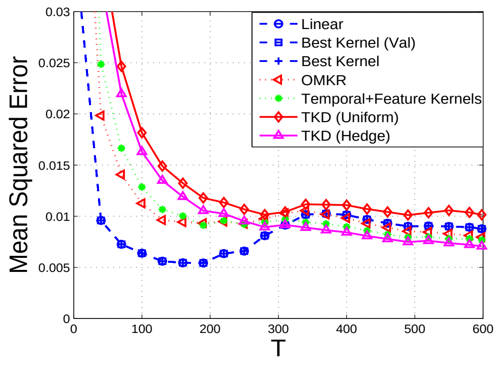
Figure 3: Mean Squared Error suffered by the algorithms as the number of instances (T) increases. For (a) and (b), the performance of Linear Kernel Regression is extremely poor, and is out of scale.



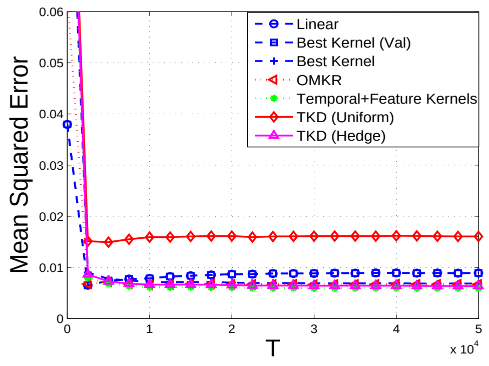
(a) Bike Demand



(b) Twitter Traffic



(c) Astrophysical (Santa Fe)



(d) Synthetic (Santa Fe)

sensitive, and validates the usage of Temporal Kernel Descriptors. TKD(Uniform) struggles to given a good performance, indicating, that a trivial combination of multiple kernels may give unstable results. Hence, it is important to learn the optimal kernel combination like we have proposed for TKD(Hedge).

The other observation is that the usage of temporal kernels itself significantly enhances the pattern recognition power as can be seen in performance of (Temporal+Feature) Kernels. This validates the power of using multi-resolution temporal kernels to measure timestamp similarity over the usage of timestamps as ordinary features with traditional kernels. Further, this performance is improved by using the temporal kernel descriptors in most cases. This validates the motivation to associate various modalities of data with multiple resolutions of time, thus giving superior performance. On the whole, it is very evident that the usage of our proposed temporal kernels and temporal kernel descriptors are significantly able to boost the performance of kernel algorithms to detect temporal patterns.

5 Conclusion

We propose temporal kernel descriptors, which overcome the existing limitations of kernel methods in finding temporal patterns by appropriately associating multi-resolution timestamps to multi-modal instance features and obtain the optimal temporal similarity between instances. Temporal kernels are devised which account for periodicity of time features. These are extended to multi-resolution temporal kernels, which measure time similarity at different resolutions. Finally, temporal kernel descriptors are developed, which find the optimal association of a multi-modal features with the different resolutions of time. The goal is to automatically learn the optimal combination of multiple kernel similarity scores. This is formulated as a Multiple Kernel Learning problem. We solve the optimization using an online learning approach. Empirically, our kernel descriptors significantly outperform traditional kernel methods due to their ability to obtain the right feature representation for the time-sensitive patterns, thus offering a novel multi-kernel view to detecting temporal patterns in the data.

References

[1] Jonathan Aflalo, Aharon Ben-Tal, Chiranjib Bhattacharyya, Jagarlapudi Saketha Nath, and Sankaran Raman. Variable sparsity kernel learning. *JMLR*, 2011.

[2] Cláudia M Antunes and Arlindo L Oliveira. Temporal data mining: An overview. In *KDD workshop on temporal data mining*, pages 1–13, 2001.

[3] Francis R Bach, Gert RG Lanckriet, and Michael I Jordan. Multiple kernel learning, conic duality, and the smo algorithm. In *Proceedings of the 21st ICML*, page 6. ACM, 2004.

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

[5] Marco Cuturi and Arnaud Doucet. Autoregressive kernels for time series. *arXiv preprint arXiv:1101.0673*, 2011.

[6] Amit Daniely, Alon Gonen, and Shai Shalev-Shwartz. Strongly adaptive online learning. *arXiv preprint arXiv:1502.07073*, 2015.

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

[8] Hadi Fanaee-T and Joao Gama. Event labeling combining ensemble detectors and background knowledge. *Progress in Artificial Intelligence*, 2(2-3):113–127, 2014.

[9] Yoav Freund and Robert 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] Yoav Freund and Robert E Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences*, 1997.

[11] Sinan Gezici, Hisashi Kobayashi, and H Vincent Poor. A new approach to mobile position tracking. In *Proc. IEEE Sarnoff Symp. Advances in Wired and Wireless Communications*, 2003.

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

[13] Steven C.H. Hoi, Rong Jin, Peilin Zhao, and Tianbao Yang. Online multiple kernel classification. *Machine Learning*, 90(2):289–316, 2013.

[14] Dirk Hovy, James Fan, Alfio Gliozzo, Siddharth Patwardhan, and Chris Welty. When did that happen?: linking events and relations to timestamps. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 185–193. Association for Computational Linguistics, 2012.

[15] Z Huang, H Chen, C Hsu, W Chen, and S Wu. Credit rating analysis with support vector machines and neural networks: a market comparative study. *DSS*, 2004.

[16] Rong Jin, S.C.H. Hoi, and Tianbao Yang. Online multiple kernel learning: Algorithms and mistake bounds. In *Algorithmic Learning Theory*. Springer Berlin Heidelberg, 2010.

[17] Vassilis Kekatos, Yu Zhang, and Georgios Giannakis. Electricity market forecasting via low-rank multi-kernel learning. *Selected Topics in Signal Processing, IEEE Journal of*, 8(6):1182–1193, 2014.

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

[19] Gert R. G. Lanckriet, Nello Cristianini, Peter Bartlett, Laurent El Ghaoui, and Michael I. Jordan. Learning the kernel matrix with semidefinite programming. *JMLR*, 5, 2004.

[20] Srivatsan Laxman and P Shanti Sastry. A survey of temporal data mining. *Sadhana*, 2006.

[21] André FT Martins, Mario AT Figueiredo, Pedro MQ Aguiar, Noah A Smith, and Eric P Xing. Online multiple kernel learning for structured prediction. 2010.

[22] Marcin Michalak. Time series prediction with periodic kernels. *Advances in Intelligent and Soft Computing*, 95:137–146, 2011.

[23] Marcin Michalak. On selection of periodic kernels parameters in time series prediction. 2014.

[24] Seyed Abolghasem Mirroshandel, Mahdy Khayyamian, and Gholamreza Ghassem-Sani. Syntactic tree kernels for event-time temporal relation learning. In *Human Language Technology. Challenges for Computer Science and Linguistics*, pages 213–223. Springer, 2011.

[25] Theophano Mitsa. *Temporal data mining*. CRC Press, 2010.

[26] Francesco Orabona, Joseph Keshet, and Barbara Caputo. The projectron: a bounded kernel-based perceptron. In *ICML*, pages 720–727. ACM, 2008.

[27] Alain Rakotomamonjy, Francis Bach, Stéphane Canu, and Yves Grandvalet. More efficiency in multiple kernel learning. In *24th ICML*, pages 775–782. ACM, 2007.

[28] Alain Rakotomamonjy, Francis Bach, Stéphane Canu, Yves Grandvalet, et al. Simplemkl. *JMLR*, 9:2491–2521, 2008.

[29] Cédric Richard, José Carlos M Bermudez, and Paul Honeine. Online prediction of time series data with kernels. *Signal Processing, IEEE Transactions on*, 57(3):1058–1067, 2009.

[30] Doyen Sahoo, Steven CH Hoi, and Bin Li. Online multiple kernel regression. In *20th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2014.

[31] Bernhard Schölkopf and Alexander J Smola. *Learning with kernels*. The MIT Press, 2002.

[32] Alex J Smola and Bernhard Schölkopf. *Learning with kernels*. Citeseer, 1998.

[33] Sören Sonnenburg, Gunnar Rätsch, Christin Schäfer, and Bernhard Schölkopf. Large scale multiple kernel learning. *JMLR*, 7:1531–1565, 2006.

[34] Francis EH Tay and Lijuan Cao. Application of support vector machines in financial time series forecasting. *Omega*, 29(4):309–317, 2001.

[35] Francis EH Tay and LJ Cao. Modified support vector machines in financial time series forecasting. *Neurocomputing*, 48(1):847–861, 2002.

[36] Zenglin Xu, Rong Jin, Irwin King, and Michael R Lyu. An extended level method for efficient multiple kernel learning. In *NIPS*, pages 1825–1832, 2008.

[37] Junyan Yang and Youyun Zhang. Application research of support vector machines in condition trend prediction of mechanical equipment. In *Advances in Neural Networks–ISNN 2005*, pages 857–864. Springer, 2005.

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