Classification of Sparse and Irregularly Sampled Time Series with Mixtures of Expected Gaussian Kernels and Random Features

Steven Cheng-Xian Li **Benjamin Marlin**
College of Information and Computer Sciences
University of Massachusetts Amherst
Amherst, MA 01003
{cxl,marlin}@cs.umass.edu

Abstract

This paper presents a kernel-based framework for classification of sparse and irregularly sampled time series. The properties of such time series can result in substantial uncertainty about the values of the underlying temporal processes, while making the data difficult to deal with using standard classification methods that assume fixed-dimensional feature spaces. To address these challenges, we propose to first re-represent each time series through the Gaussian process (GP) posterior it induces under a GP regression model. We then define kernels over the space of GP posteriors and apply standard kernel-based classification. Our primary contributions are (i) the development of a kernel between GPs based on the mixture of kernels between their finite marginals, (ii) the development and analysis of extensions of random Fourier features for scaling the proposed kernel to large-scale data, and (iii) an extensive empirical analysis of both the classification performance and scalability of our proposed approach.

1 INTRODUCTION

In this paper, we address the problem of classification of sparse and irregularly sampled time series. Irregularly sampled (or non-uniformly sampled) time series are characterized by variable time intervals between successive observations. While all time series in a data set are typically defined on the same continuous-time interval, the number of observations per time series can vary. When the intervals between successive observations are long, the time series are said to be sparsely sampled.

Such time series data arise when sampling complex temporal processes in a number of important areas including climate science [Schulz and Stattegger, 1997], ecology [Clark and Bjørnstad, 2004], biology [Ruf, 1999], medicine [Marlin et al., 2012] and astronomy [Scargle, 1982]. In domains including medicine, the data are both irregularly sampled and sparsely sampled [Marlin et al., 2012]. Classification in this setting is challenging both because the data cases are not naturally defined in a fixed-dimensional feature space due to irregular sampling and variable time series length, and because there can be substantial uncertainty about the underlying temporal processes due to the sparsity of observations.

To address these challenges, we begin by re-representing each input time series using the Gaussian process (GP) posterior it induces under a GP regression model [Rasmussen and Williams, 2006]. We then define kernels over the space of GP posteriors and apply standard kernel-based classification methods [Cortes and Vapnik, 1995]. We propose a kernel between GPs based on the mixture of kernels between finite dimensional GP marginal distributions defined over sliding time windows. We refer to this as the *mixture of sliding GP marginal kernels (MSM)* framework.

The MSM kernel framework requires a base kernel between finite-dimensional GP marginals, which are Gaussian distributions. While the MSM framework can be used with any valid base kernel between two Gaussian distributions, we propose an uncertainty-aware base kernel that we refer to as the *expected Gaussian kernel* to address the uncertainty that results from sparse sampling. Using the expected Gaussian kernel in the MSM framework yields a kernel between Gaussian processes that we refer to as the *mixture of expected Gaussian kernels (MEG)*.

Next, we consider the problem of scaling our proposed kernel-based time series classification framework to large-scale data. Computing the exact Gram matrix for the proposed MEG kernel with n time series takes $\mathcal{O}(n^2d^3k)$ time when k sliding windows of length d are used. To address this problem, we show how to extend the random Fourier feature kernel approximation framework [Rahimi and Recht, 2007] to the MEG kernel. Using our random feature construction, it takes $\mathcal{O}(nMd^2)$ time to compute the random feature matrix using M random features for

each time series. We then show how to use Fastfood [Le et al., 2013] to reduce the random feature computation time to $\mathcal{O}(nMd \log d)$ when the window size d is large. With an extra rank-r covariance approximation, we can further reduce this time to $\mathcal{O}(nMr \log d)$ for $r \ll d$. Finally, we provide a convergence analysis of our proposed approximation based on the recently developed matrix Bernstein inequality [Lopez-Paz et al., 2014; Tropp, 2012a].

The primary contributions of this paper are:

1. The development of an uncertainty-aware kernel for GPs based on the mixture of kernels between their finite-dimensional marginals (Section 3).
2. The development and analysis of an extension of random Fourier features for scaling the proposed kernel to large-scale data (Section 4).
3. An extensive empirical analysis of both the classification performance and scalability of our proposed approach (Section 5).

In Section 2, we begin by describing how to represent sparse and irregularly sampled time series using Gaussian processes, which are a fundamental building block of our proposed framework.

2 SPARSE AND IRREGULARLY SAMPLED TIME SERIES

Our focus in this paper is classification of time series data in the presence of sparse and irregular sampling. Consider a data set of n independent time series $\mathcal{D} = \{\mathcal{S}_1, \ldots, \mathcal{S}_n\}$, where each time series $\mathcal{S}_i$ is represented as a list of time points $\mathbf{t}_i = [t_{i1}, \ldots, t_{i|\mathcal{S}_i|}]^\top$, and a list of corresponding values $\mathbf{y}_i = [y_{i1}, \ldots, y_{i|\mathcal{S}_i|}]^\top$. We assume that each time series is defined over a common continuous-time interval $[0, T]$. However, for irregularly sampled time series we do not assume that all of the time series are defined on the same collection of time points (i.e., $\mathbf{t}_i \neq \mathbf{t}_j$ in general), we do not assume that the intervals between time points are uniform, and we do not assume that the number of observations in different time series is the same (i.e., $|\mathcal{S}_i| \neq |\mathcal{S}_j|$ in general).

Learning in this setting is challenging because the data cases are not naturally defined in a fixed-dimensional feature space due to irregular sampling, and there can be substantial uncertainty about the underlying temporal processes due to the sparsity of observations.

To address these challenges, we begin by re-representing each input time series using the Gaussian process (GP) posterior it induces under a Gaussian process regression model [Rasmussen and Williams, 2006]. This construction naturally accommodates sparse and irregularly sampled time series. To obtain the posterior GPs, we use a Gaussian likelihood function with noise variance σ^2, and a zero-mean GP prior with the squared exponential covariance function

$$\mathcal{K}_{\text{SE}}(t_i, t_j) = a \exp(-b(t_i - t_j)^2), \text{ for } a, b > 0.$$

We learn the hyperparameters a, b, σ of the GP regression model by maximizing the marginal likelihood of the data.[1] Under this model, the posterior distribution induced by each time series $\mathcal{S}$ is also a Gaussian process. By definition, any finite marginal of a GP is Gaussian distributed. Let $\mathcal{GP}(\mathbf{u} \mid \mathcal{S} = \{\mathbf{t}, \mathbf{y}\}) = \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ denote the posterior GP marginal of $\mathcal{S}$ over a collection of time points $\mathbf{u}$. The mean and covariance of the Gaussian posterior marginal $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ is given below where $[K(\mathbf{u}, \mathbf{t})]_{ij} = \mathcal{K}_{\text{SE}}(u_i, t_j)$.

$$\begin{aligned} \boldsymbol{\mu} &= K(\mathbf{u}, \mathbf{t}) \left(K(\mathbf{t}, \mathbf{t}) + \sigma^2 \mathbf{I}\right)^{-1} \mathbf{y}, \\ \boldsymbol{\Sigma} &= K(\mathbf{u}, \mathbf{u}) - K(\mathbf{u}, \mathbf{t}) \left(K(\mathbf{t}, \mathbf{t}) + \sigma^2 \mathbf{I}\right)^{-1} K(\mathbf{t}, \mathbf{u}). \end{aligned} \tag{1}$$

Applying GP regression requires $\mathcal{O}(|\mathcal{S}|^3)$ time due to the matrix inversion in (1). We note that efficient approximation algorithms are available when working with long time series [Hensman et al., 2013; Quiñonero-Candela and Rasmussen, 2005].

In the next section, we describe our proposed framework for defining kernels between time series based on Gaussian processes.

3 KERNELS FOR TIME SERIES

In this section, we first introduce the general mixture of sliding GP marginal kernels (MSM) framework for sparse and irregularly sampled data. We then introduce the expected Gaussian kernel, which serves as an uncertainty-aware base kernel within the MSM framework.

3.1 THE MIXTURE OF SLIDING GP MARGINAL KERNELS

As described in Section 2, we represent each time series $\mathcal{S}$ in a data set $\mathcal{D}$ using the posterior Gaussian process it induces under a GP regression model. The proposed mixture of sliding GP marginal kernels $\mathcal{K}^{(d)}_{\text{MSM}}$ defines a kernel between a pair of time series through a weighted average of a base kernel $\mathcal{K}_{\text{B}}$ applied to a collection of finite posterior GP marginals. Specifically, let $u_1, \ldots, u_L$ be a uniformly-spaced set of L time points on $[0, T]$, and $\mathbf{u}^{(s)} = [u_s, \ldots, u_{s+d-1}]^\top$ be a window of d time points starting at u_s. The MSM kernel compares the posterior GP marginals over the complete collection of valid sliding windows $\mathbf{u}^{(1)}, ..., \mathbf{u}^{L-d+1}$ as shown below, provided $w_s \geq 0$ for all s.

$$\mathcal{K}^{(d)}_{\text{MSM}}(\mathcal{S}_i, \mathcal{S}_j) = \sum_{s=1}^{L-d+1} w_s \mathcal{K}_{\text{B}}\Big(\mathcal{GP}(\mathbf{u}^{(s)} | \mathcal{S}_i), \mathcal{GP}(\mathbf{u}^{(s)} | \mathcal{S}_j)\Big)$$

[1] See Rasmussen and Williams [2006] for details.

The length of the windows d is a hyper-parameter of the MSM kernel. In this work, we choose uniform kernel mixture weights $w_s = 1/k$. Alternatively, the kernel weights can be learned from data using multiple kernel learning algorithms [Bach et al., 2004].

The base kernel $\mathcal{K}_{\text{B}}$ can be any valid kernel that takes as input two d-dimensional Gaussians. Of particular interest are uncertainty-aware base kernels that use the covariance information in the posterior marginals to modulate the similarity between the distributions. We present an uncertainty-aware expected Gaussian kernel in Section 3.3, but first describe a simpler kernel to highlight the trade-offs induced by the window length parameter d.

3.2 GAUSSIAN KERNEL ON MARGINAL MEANS

The Gaussian kernel $\mathcal{K}_{\text{G}}$ below is one of the most widely used kernels in machine learning.

$$\mathcal{K}_{\text{G}}(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\frac{1}{2\gamma^2}\|\mathbf{x}_i - \mathbf{x}_j\|^2\right) \tag{2}$$

The parameter γ controls the bandwidth of the kernel. The Gaussian kernel $\mathcal{K}_{\text{G}}$ provides a simple kernel $\mathcal{K}_{\text{G}\mu}$ between Gaussian distributions $\mathcal{N}_i = \mathcal{N}(\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)$ and $\mathcal{N}_j = \mathcal{N}(\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)$ when applied to their mean vectors as follows.

$$\mathcal{K}_{\text{G}\mu}(\mathcal{N}_i, \mathcal{N}_j) = \mathcal{K}_{\text{G}}(\boldsymbol{\mu}_i, \boldsymbol{\mu}_j) \tag{3}$$

Importantly, this kernel is not uncertainty aware as it discards the covariances from the posterior Gaussians. We use the notation $\mathcal{K}^{(d)}_{\text{MG}}$ to denote the use of $\mathcal{K}_{\text{G}\mu}$ as the base kernel within the MSM framework. In the case where $d = 1$, the $\mathcal{K}^{(d)}_{\text{MG}}$ kernel corresponds to taking the average similarity between the means of the two marginal posterior distributions as seen below.

$$\mathcal{K}^{(1)}_{\text{MG}}(\mathcal{S}_i, \mathcal{S}_j) = \frac{1}{L}\sum_{s=1}^{L}\exp\left(-\frac{1}{2\gamma^2}(\mu_{is} - \mu_{js})^2\right)$$

On the other hand, when $d = L$, the $\mathcal{K}^{(d)}_{\text{MG}}$ kernel is equivalent to a product of the similarities between the means of the two marginal posterior distributions as seen below.

$$\begin{aligned}\mathcal{K}^{(L)}_{\text{MG}}(\mathcal{S}_i, \mathcal{S}_j) &= \exp\left(-\frac{1}{2\gamma^2}\sum_{s=1}^{L}(\mu_{is} - \mu_{js})^2\right) \\ &= \prod_{s=1}^{L}\exp\left(-\frac{1}{2\gamma^2}(\mu_{is} - \mu_{js})^2\right)\end{aligned}$$

This comparison shows that $\mathcal{K}^{(1)}_{\text{MG}}$ is much more likely to be robust to the influence of noise and outliers due to the use of averaging, but it ignores the broader structure across time points. On the other hand, $\mathcal{K}^{(L)}_{\text{MG}}$ captures the broader structure across time points, but may be more sensitive to the presence of noise and outliers due to the product from of the kernel. Importantly, the MSM kernel framework is able to balance these considerations by allowing for the selection of intermediate window lengths d.

3.3 THE EXPECTED GAUSSIAN KERNEL

In this section, we present an uncertainty-aware base kernel $\mathcal{K}_{\text{EG}}$, which we refer to as the *expected Gaussian kernel*. This kernel is obtained as the expectation of the standard Gaussian kernel shown in (2) under the two independent Gaussians $\mathcal{N}_i$ and $\mathcal{N}_j$

$$\mathcal{K}_{\text{EG}}(\mathcal{N}_i, \mathcal{N}_j) = \mathbb{E}_{\mathbf{x}_i\sim\mathcal{N}_i, \mathbf{x}_j\sim\mathcal{N}_j}\left[\mathcal{K}_{\text{G}}(\mathbf{x}_i, \mathbf{x}_j)\right].$$

Importantly, the value of the expected Gaussian kernel can be computed analytically as shown in (4) where $\widetilde{\boldsymbol{\mu}} = \boldsymbol{\mu}_i - \boldsymbol{\mu}_j$ and $\widetilde{\boldsymbol{\Sigma}} = \boldsymbol{\Sigma}_i + \boldsymbol{\Sigma}_j + \gamma^2\mathbf{I}$. (see Appendix A for the derivation).

$$\mathcal{K}_{\text{EG}}(\mathcal{N}_i, \mathcal{N}_j) = \sqrt{\frac{|\boldsymbol{\Sigma}|}{|\widetilde{\boldsymbol{\Sigma}}|}}\exp\left(-\frac{1}{2}\widetilde{\boldsymbol{\mu}}^\top\widetilde{\boldsymbol{\Sigma}}^{-1}\widetilde{\boldsymbol{\mu}}\right). \tag{4}$$

The positive definiteness of the expected Gaussian kernel follows from the fact that the Gaussian kernel is positive definite and therefore there exists a map $\boldsymbol{\phi}$ such that the kernel acts as a dot product $\langle\boldsymbol{\phi}(\mathbf{x}_i), \boldsymbol{\phi}(\mathbf{x}_j)\rangle$. With the independence assumption, the expected Gaussian kernel also acts as a dot product over the expected map [Smola et al., 2007].

$$\begin{aligned}\mathcal{K}_{\text{EG}}(\mathcal{N}_i, \mathcal{N}_j) &= \mathbb{E}_{\mathbf{x}_i\sim\mathcal{N}_i, \mathbf{x}_j\sim\mathcal{N}_j}\langle\boldsymbol{\phi}(\mathbf{x}_i), \boldsymbol{\phi}(\mathbf{x}_j)\rangle \\ &= \langle\mathbb{E}_{\mathbf{x}\sim\mathcal{N}_i}[\boldsymbol{\phi}(\mathbf{x})], \mathbb{E}_{\mathbf{x}\sim\mathcal{N}_j}[\boldsymbol{\phi}(\mathbf{x})]\rangle.\end{aligned}$$

Interestingly, the probability product kernel of Jebara et al. [2004] applied to a pair of Gaussian distributions

$$\mathcal{K}_{\text{PP}}(\mathcal{N}_i, \mathcal{N}_j) = \int \mathcal{N}(\mathbf{x}; \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)^\rho \mathcal{N}(\mathbf{x}; \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)^\rho \, \mathrm{d}\mathbf{x}$$

when $\rho = 1$ (also known as the expected likelihood kernel) is a limiting case of the expected Gaussian kernel as $\gamma \to 0$. In this case, the $\mathcal{K}_{\text{G}}$ term inside $\mathcal{K}_{\text{EG}}$ degenerates to the Dirac delta function $\delta(\mathbf{x}_i - \mathbf{x}_j)$, and the expected Gaussian kernel collapses to the probability product kernel with $\rho = 1$ by the sifting property of the delta function.

We refer to the use of the expected Gaussian kernel within the MSM framework as the *mixture of expected Gaussian kernels (MEG)*. Similar to $\mathcal{K}^{(d)}_{\text{MG}}$, the MEG kernel is able to strike a balance between the use of averaging to mitigate noise and the use of higher-dimensional marginals to capture broader temporal structure under uncertainty through the choice of d.

In terms of computational complexity, computing the expected Gaussian kernel (4) for d-dimensional Gaussians takes $\mathcal{O}(d^3)$ time because of the inversion of $\widetilde{\boldsymbol{\Sigma}}$ and the

computation of its determinant. As a result, for the MEG kernel involving k GP marginals of d dimensions, it takes $\mathcal{O}(kn^2d^3)$ time to compute the $n \times n$ kernel matrix over n data cases. In the next section, we discuss scaling learning with MEG kernels to large data sets using random feature approximations.

4 RANDOM FOURIER FEATURES

The $\mathcal{O}(n^2)$ kernel matrix computation time is a significant limitation when working with large data sets. Random Fourier feature approximation [Rahimi and Recht, 2007] is a kernel approximation algorithm based on Bochner's theorem that maps the input data into a randomized low-dimensional feature space to approximate a shift-invariant kernel. In this section, we show how to extend this idea to scale-up learning with expected Gaussian kernels and apply the result to the MEG kernel.

4.1 RANDOM FEATURES FOR EXPECTED GAUSSIAN KERNELS

Following the construction presented by Rahimi and Recht [2007], the Gaussian kernel $\mathcal{K}_{\text{G}}$ defined in (2) can be approximated by an m-dimensional random vector

$$\mathbf{z}(\mathbf{x}) = \sqrt{\frac{2}{m}}\big[\cos(\mathbf{w}_1^\top \mathbf{x} + b_1), \ldots, \cos(\mathbf{w}_m^\top \mathbf{x} + b_m)\big]^\top,$$

where $\mathbf{w}_i \sim \mathcal{N}(\mathbf{0}, \gamma^{-2}\mathbf{I})$,[2] and $b_i \sim \text{uniform}(0, 2\pi)$ so that $\mathcal{K}_{\text{G}}(\mathbf{x}_i, \mathbf{x}_j) \approx \mathbf{z}(\mathbf{x}_i)^\top \mathbf{z}(\mathbf{x}_j)$.

The analytic form of the expected Gaussian kernel given in (4) is not shift invariant in terms of the means and covariances of the input Gaussians, and therefore we cannot directly expand the kernel as in Rahimi and Recht [2007]. However, we can derive the random Fourier features for the expected Gaussian kernel by taking the expectation after Gaussian kernel expansion. With the independence of the input Gaussians, we have

$$\begin{aligned}\mathbb{E}_{\mathbf{x}_i\mathbf{x}_j}[\mathcal{K}_{\text{G}}(\mathbf{x}_i, \mathbf{x}_j)] &\approx \mathbb{E}_{\mathbf{x}_i\mathbf{x}_j}\left[\mathbf{z}(\mathbf{x}_i)^\top \mathbf{z}(\mathbf{x}_j)\right] \\ &= \mathbb{E}_{\mathbf{x}_i}[\mathbf{z}(\mathbf{x}_i)]^\top \mathbb{E}_{\mathbf{x}_j}[\mathbf{z}(\mathbf{x}_j)].\end{aligned}$$

Next, we note that each entry of $\mathbb{E}_{\mathbf{x}\sim\mathcal{N}(\boldsymbol{\mu},\boldsymbol{\Sigma})}[\mathbf{z}(\mathbf{x})]$ can be obtained analytically as shown below. This result exploits the fact that the expectation of the complex random feature map $\exp(i\mathbf{w}^\top\mathbf{x})$ derived from the Fourier expansion of the kernel function is the characteristic function of the distribution of $\mathbf{x}$. A detailed derivation is given in Appendix B.

$$\begin{aligned}\mathbb{E}[z_i(\mathbf{x})] &= \sqrt{\frac{2}{m}}\,\mathbb{E}_{\mathbf{x}\sim\mathcal{N}(\boldsymbol{\mu},\boldsymbol{\Sigma})}[\cos(\mathbf{w}_i^\top\mathbf{x} + b_i)] \\ &= \sqrt{\frac{2}{m}}\,\exp\left(-\frac{1}{2}\mathbf{w}_i^\top\boldsymbol{\Sigma}\mathbf{w}_i\right)\cos(\mathbf{w}_i^\top\boldsymbol{\mu} + b_i).\end{aligned}$$

[2] $\mathbf{w}_i \sim \mathcal{N}(\mathbf{0}, \gamma^{-2}\mathbf{I})$ can be done with each entry drawn independently from $\mathcal{N}(0, \gamma^{-2})$.

Algorithm 1: Random Fourier Features for $\mathcal{K}_{\text{EG}}$

Input: A Gaussian $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ with mean $\boldsymbol{\mu}$ and covariance $\boldsymbol{\Sigma}$. Width parameter γ^2 of the Gaussian kernel. Number of random features m.

$\mathbf{w}_1, \ldots, \mathbf{w}_m \overset{\text{iid}}{\sim} \mathcal{N}(\mathbf{0}, \gamma^{-2}\mathbf{I})$

$b_1, \ldots, b_m \overset{\text{iid}}{\sim} \text{uniform}(0, 2\pi)$

return $\sqrt{\frac{2}{m}}\begin{bmatrix}\exp\left(-\frac{1}{2}\mathbf{w}_1^\top\boldsymbol{\Sigma}\mathbf{w}_1\right)\cos(\mathbf{w}_1^\top\boldsymbol{\mu} + b_1) \\ \vdots \\ \exp\left(-\frac{1}{2}\mathbf{w}_m^\top\boldsymbol{\Sigma}\mathbf{w}_m\right)\cos(\mathbf{w}_m^\top\boldsymbol{\mu} + b_m)\end{bmatrix}$

As we can see, each random feature for an expected Gaussian kernel is the product of $\sqrt{2/m}\cos(\mathbf{w}_i^\top\boldsymbol{\mu} + b_i)$ and $\exp\left(-\frac{1}{2}\mathbf{w}_i^\top\boldsymbol{\Sigma}\mathbf{w}_i\right)$. The former is identical to the random Fourier feature with the Gaussian mean as input. The latter is an exponential decay term that decreases as uncertainty in the distribution increases.

The complete procedure for obtaining a random features approximation for the expected Gaussian kernel is given in Algorithm 1.

For the d-dimensional case, approximating an expected Gaussian kernel with m random features using Algorithm 1 requires $\mathcal{O}(md^2)$ time as it takes $\mathcal{O}(d^2)$ time to compute the quadratic term $\mathbf{w}^\top\boldsymbol{\Sigma}\mathbf{w}$. As a result, given a data set of size n, it takes $\mathcal{O}(nmd^2)$ to compute the $n \times m$ feature matrix. This is more efficient compared to the $\mathcal{O}(n^2d^3)$ time needed to compute the exact kernel, especially when $n \gg m$.

4.2 ACCELERATION FOR HIGH-DIMENSIONAL GAUSSIANS

The $\mathcal{O}(d^2)$ time for computing the random features can be computationally prohibitive when working with high-dimensional Gaussians. Le et al. [2013] proposed Fastfood to accelerate the computation of the original random Fourier features of Rahimi and Recht [2007]. Fastfood utilizes the fast Walsh-Hadamard transform to simulate a full Gaussian random matrix using a small portion of i.i.d. Gaussian samples. Essentially, given a vector $\mathbf{x} \in \mathbb{R}^d$, Fastfood approximates the matrix-vector product $\mathbf{V}\mathbf{x}$ in $\mathcal{O}(m \log d)$ time instead of $\mathcal{O}(md)$, where $\mathbf{V}$ is an $m \times d$ random matrix with each entry drawn independently from $\mathcal{N}(0, \gamma^{-2})$.

With Fastfood, computing the $\cos(\mathbf{w}_i^\top\boldsymbol{\mu} + b_i)$ term for the expected Gaussian kernel for all $i = 1, \ldots, m$ can be done by first generating the random vector $\mathbf{V}\boldsymbol{\mu}$ as described above. All m entries $\cos([\mathbf{V}\boldsymbol{\mu}]_i + b_i)$ can then be computed in $\mathcal{O}(m \log d)$ time.

The bottleneck for the expected Gaussian kernel is the computation of the exponential term $\exp\left(-\frac{1}{2}\mathbf{w}_i^\top\boldsymbol{\Sigma}\mathbf{w}_i\right)$, which needs $\mathcal{O}(d^2)$ time if computed naively. This can also be

accelerated by using the Fastfood trick twice:

$$\exp\left(-\frac{1}{2}\mathbf{w}_i^\top \boldsymbol{\Sigma}\mathbf{w}_i\right) = \exp\left(-\frac{1}{2}[\mathbf{V}(\mathbf{V}\boldsymbol{\Sigma})^\top]_{ii}\right). \quad (5)$$

Following the stacking strategy in Le et al. [2013], we choose $\mathbf{V}$ in (5) to be a $d \times d$ square matrix[3], and repeat this step $\lceil m/d \rceil$ times to produce all m features. This leads to an overall cost of $\mathcal{O}(md \log d)$ as opposed to the $\mathcal{O}(md^2)$ time mentioned before to compute m random features for the expected Gaussian kernel taking on a single d-dimensional Gaussian input.

We can further reduce the cost by approximating the covariance matrix with a low rank matrix $\boldsymbol{\Sigma} \approx \boldsymbol{\Phi}\boldsymbol{\Phi}^\top$ using truncated SVD where $\boldsymbol{\Phi} \in \mathbb{R}^{d\times r}$. There exists efficient algorithms to compute top-r SVD such as randomized SVD [Halko et al., 2011] that requires $\mathcal{O}(d^2 \log r)$ time in contrast with $\mathcal{O}(d^3)$ for classical algorithms. With Fastfood, the exponential term can be approximated in $\mathcal{O}(r \log d)$ time:

$$\exp\left(-\frac{1}{2}\mathbf{w}_i^\top \boldsymbol{\Sigma}\mathbf{w}_i\right) \approx \exp\left(-\frac{1}{2}\sum_{j=1}^{r}[\mathbf{V}\boldsymbol{\Phi}]_{ij}^2\right). \quad (6)$$

This leads to $\mathcal{O}(mr \log d)$ time for some $r < d$ to compute m random features for a single data case.

4.3 RANDOM FEATURES FOR THE MIXTURE OF EXPECTED GAUSSIAN KERNELS

Let $\mathbf{z}(\mathcal{N})$ denote the random features for the expected Gaussian kernel computed by Algorithm 1. As described in Section 3.1, each time series $\mathcal{S}_i$ is summarized by a collection of k Gaussian marginals $\{\mathcal{N}_i^{(1)}, \ldots, \mathcal{N}_i^{(k)}\}$ for all i. The mixture of expected Gaussian kernels can be approximated by the random features of the base kernel applied to each marginal:

$$\sum_{s=1}^{k} w_s \mathcal{K}_{\text{EG}}(\mathcal{N}_i^{(s)}, \mathcal{N}_j^{(s)}) \approx \sum_s w_s \mathbf{z}_s(\mathcal{N}_i^{(s)})^\top \mathbf{z}_s(\mathcal{N}_j^{(s)}) = \sum_s \left(\sqrt{w_s}\mathbf{z}_s(\mathcal{N}_i^{(s)})\right)^\top \left(\sqrt{w_s}\mathbf{z}_s(\mathcal{N}_j^{(s)})\right).$$

Equivalently, each time series can be expressed as the compound random feature map below to approximate the MEG kernel.

$$\widehat{\mathbf{z}}(\mathcal{S}) = \left[\sqrt{w_1}\mathbf{z}_1(\mathcal{N}^{(1)})^\top, \ldots, \sqrt{w_k}\mathbf{z}_k(\mathcal{N}^{(k)})^\top\right]^\top. \quad (7)$$

In this work, we set $w_s = 1/k$ for all k marginals, that is, the base kernels are weighted equally. Furthermore, each marginal $\mathcal{N}^{(s)}$ is approximated by the same number of random features m. Therefore, $\widehat{\mathbf{z}}(\mathcal{S})$ has mk random features in total. In Section 4.4, we will show that having the same number of random features for each marginal will lead to the lowest error bound under uniform weights.

In general, w_s can be the coefficients of any non-negative combination, either chosen according to domain knowledge or learned from data. Learning the weights from data with the random features given in (7) can be viewed as an approximation to multiple kernel learning [Bach et al., 2004]. Optimizing $w_1, \ldots, w_k$ is similar to Mahalanobis metric learning [Xing et al., 2002] for the diagonal case except that all random features come from the same base kernel share a scaling factor.

4.4 APPROXIMATION GUARANTEES

In this section, we analyze the approximation quality of the expected Gaussian kernel random features computed by Algorithm 1 in terms of the concentration of the approximating kernel matrix. Using the Hermitian matrix Bernstein inequality [Tropp, 2012a,b] and following a derivation similar to [Lopez-Paz et al., 2014], we can bound the spectral norm (denoted $\|\cdot\|$) of the difference between the exact expected Gaussian kernel and its approximation.[4]

Theorem 1. *Given a data set with each example represented as a single Gaussian, $\mathcal{N}_1, \ldots, \mathcal{N}_n$, let $\mathbf{K} \in \mathbb{R}^{n\times n}$ be the expected Gaussian kernel matrix. Let $\widehat{\mathbf{K}} \in \mathbb{R}^{n\times n}$, with each entry $[\widehat{\mathbf{K}}]_{ij} = \mathbf{z}(\mathcal{N}_i)^\top \mathbf{z}(\mathcal{N}_j)$, be the approximation matrix constructed using Algorithm 1 with m random features. Then we have*

$$\mathbb{E}\|\widehat{\mathbf{K}} - \mathbf{K}\| \leq \frac{2n}{m}\sqrt{\frac{2\log n}{m}} + \frac{2n\log n}{3m^2}.$$

The proof of Theorem 1 is given in Appendix C. It states that the error $\mathbb{E}\|\widehat{\mathbf{K}} - \mathbf{K}\|$ is bounded by $\mathcal{O}(n \log n)$ for n data cases with a fixed number of random features m. On the other hand, for a fixed number of data cases n, increasing the number of random features m induces an $\mathcal{O}(m^{-3/2})$ reduction in error.

As for the high-dimensional case described in Section 4.2, Le et al. [2013] have shown that the Fastfood feature map is unbiased, and therefore Theorem 1 also holds for the random features computed by Fastfood using (5). However, with the low-rank approximation used in (6), $\widehat{\mathbf{K}}$ no longer converges to $\mathbf{K}$ but instead converges to $\widetilde{\mathbf{K}}$ where

$$[\widetilde{\mathbf{K}}]_{ij} = \mathcal{K}_{\text{EG}}\left(\mathcal{N}(\boldsymbol{\mu}_i, \boldsymbol{\Phi}_i\boldsymbol{\Phi}_i^\top), \mathcal{N}(\boldsymbol{\mu}_j, \boldsymbol{\Phi}_j\boldsymbol{\Phi}_j^\top)\right).$$

Following the construction described in Section 4.3, the mixture of k expected Gaussian kernels has a total of $M = |\widehat{\mathbf{z}}(\mathcal{S})| = mk$ features. Since $w_s = 1/k$ for all s, each entry of the feature vector is in the form of

[3] Assume $\boldsymbol{\Sigma}$ is properly padded so that the dimension becomes $d = 2^\ell$ in order to perform Hadamard transform [Le et al., 2013].

[4] This bound can also be applied to the original random Fourier feature approximation [Rahimi and Recht, 2007].

$\sqrt{2/M}\,\mathbb{E}[\cos(\mathbf{w}^\top \mathbf{x}+b)]$, whose absolute value is bounded by $\sqrt{2/M}$. Following the proof of Theorem 1, we can bound the error of using the proposed random feature approximation to the MEG kernel.

Corollary 1. *Consider the MEG kernel consisting of k base kernels. Let $\mathbf{K} \in \mathbb{R}^{n\times n}$ be the MEG kernel matrix, and $\widehat{\mathbf{K}}$ be the approximating matrix using $M = mk$ random features. Then,*

$$\mathbb{E}\|\widehat{\mathbf{K}} - \mathbf{K}\| \leq \frac{2n}{M}\sqrt{\frac{2\log n}{M}} + \frac{2n\log n}{3M^2}. \qquad (8)$$

The expected error bound in Corollary 1 has the same form as that in Theorem 1 except the bound is determined by the number of total random features M. When the number of kernels k is large, even if each kernel is approximated by only a few random features, a low error bound can still be achieved if $M = mk$ is sufficiently large.

As a matter of fact, for a *convex combination* of k expected Gaussian kernels with unequal weights, choosing the number of random features proportional to the corresponding kernel weight will achieve an error bound identical to (8) for a total of M random features.

5 EXPERIMENTS

We evaluate the proposed MEG kernel and the corresponding random feature approximation in terms of time series classification in the presence of sparse and irregular sampling. In the sections below, we describe the experimental methodology and the results.

5.1 EXPERIMENTAL METHODOLOGY

Data. We conduct experiments on all 43 time series data sets from the UCR time series classification archive [Keogh et al., 2011]. The UCR archive contains a diverse collection of time series data sets that vary significantly in terms of length, number of classes, and number of data cases. However, all the data sets are densely and uniformly sampled. This allows us to perform controlled experiments where we decrease the sampling density and observe the effect on the relative performance of different classification methods. We consider ten different sampling densities from 10% to 100% in steps of 10%.

Gaussian Process Representation. Following Section 2, for each data set and each sampling density, we first learn the hyperparameters of the GP regression model by optimizing the log marginal likelihood over the observed data. As described in Section 3.1, we compute the posterior GP marginals over a uniformly-spaced grid of L points on $[0, T]$, where T is the common length of the fully observed time series of each data set. We select $L = \min(3T, 500)$.

Kernel and Feature Normalization. We apply standard kernel normalization to all kernel matrices before averaging. We also normalize each random feature vector to have unit length. Empirically, we found that normalization improves the classification performance.

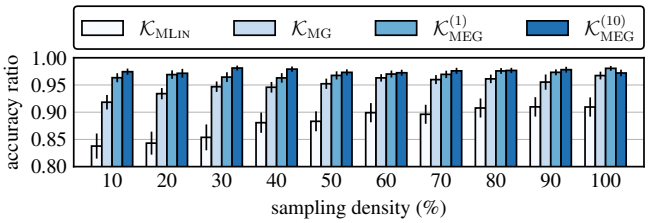


Figure 1: Comparing MSM kernel framework with different base kernels.

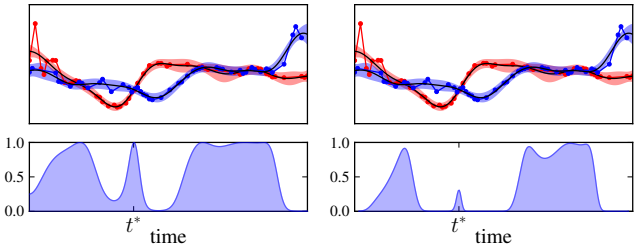
Base Classifiers and Hyperparameters. We use support vector machines (SVMs) for classification. For kernel-based methods we use libsvm [Chang and Lin, 2011] with precomputed kernels. For random feature approximation, we use liblinear [Fan et al., 2008], which is tailored for linear models. We use five-fold stratified cross validation to jointly select the SVM regularization parameter and the parameter γ for the expected Gaussian kernels.

Accuracy Measures. We assess the performance of each method in terms of classification accuracy using the benchmark train/test splits in the UCR archive. We report results in terms of average *accuracy ratios* over all 43 data sets to emphasize the relative differences between methods across different sampling densities. For a given data set and sampling density, the accuracy ratio for a method is the accuracy of the method divided by the accuracy of the best performing method on that data set and sampling density. We also report one-standard-error error bars.

5.2 EXPERIMENTS AND RESULTS

Comparing Base Kernels for MSM Framework. We evaluate several instances of the MSM framework using different base kernels. The linear MSM kernel $\mathcal{K}_{\text{MLIN}}$ uses the linear kernel on the univariate marginal means $\mathcal{K}_{\text{LIN}}(\mathcal{N}_i, \mathcal{N}_j) = \mu_i\mu_j$ as the base kernel. The Gaussian MSM kernel $\mathcal{K}_{\text{MG}}$ uses $\mathcal{K}_{\text{G}\mu}$ defined in (3) also on the univariate marginal means. We compare these baseline methods to two expected Gaussian kernel based MSM kernels: the MEG kernel $\mathcal{K}^{(1)}_{\text{MEG}}$ on the univariate marginals, and the MEG kernel $\mathcal{K}^{(10)}_{\text{MEG}}$ with a sliding window size of 10.

Figure 1 shows the classification performance of these methods on each sampling density. The Gaussian MSM kernel $\mathcal{K}_{\text{MG}}$ significantly outperforms the linear MSM kernel $\mathcal{K}_{\text{MLIN}}$. However, $\mathcal{K}^{(1)}_{\text{MEG}}$ and $\mathcal{K}^{(10)}_{\text{MEG}}$ both outperform $\mathcal{K}_{\text{MG}}$, particularly under high sparsity. This is expected since $\mathcal{K}^{(1)}_{\text{MEG}}$ and $\mathcal{K}^{(10)}_{\text{MEG}}$ both capture posterior uncertainty while $\mathcal{K}_{\text{MG}}$ does not.



(a) window size $d = 1$ (b) window size $d = 10$

Figure 2: Comparison of different window size d. The plot on the top of each panel shows two time series from the ECG200 data set at 50% sampling density with visualization of their posterior Gaussian process. The plot on the bottom shows the value of the corresponding expected Gaussian kernel at each time slice.

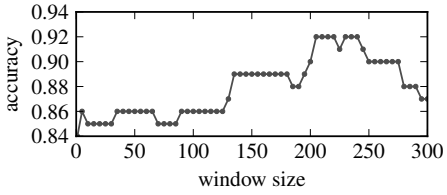


Figure 3: Comparison of classification accuracy under different window sizes on ECG200 at 50% sampling density.

Effect of Sliding Window Size for MEG Kernel. Figure 2 illustrates how different window sizes affect the similarity output by the expected Gaussian kernel on the ECG200 data set from the UCR archive at 50% sampling density. The two time series intersect at around t^*; however, they have opposite trends at this time point. Since $\mathcal{K}_{\text{MEG}}^{(1)}$ does not take local correlation structure into account at all, it achieves the highest possible value at t^*. On the other hand, $\mathcal{K}_{\text{MEG}}^{(10)}$ outputs a low value at t^* since a larger window captures the fact that the two processes are anti-correlated in the neighborhood of t^*.

Figure 3 shows the classification performance across various window sizes ranging from 1 to L on the ECG200 data set at 50% sampling density. For this experiment, we fixed the SVM regularization parameter to $C = 2000$, and the covariance parameter of the expected Gaussian kernel to $\gamma = 0.01d$, which grows linearly as the window size d increases. Empirically, such choice of γ makes the values of the expected Gaussian kernels numerically stable.

The results show that the classification accuracy on ECG200 improves as the window size increases and peaks at around $0.75L$. We note that using larger window size is computationally more expensive, and that not all data sets show a benefit with increasing window size.

Comparing MEG to Existing Methods. We compare the MEG kernel with two existing methods for time series classification. The reproducing kernel Hilbert space (RKHS)

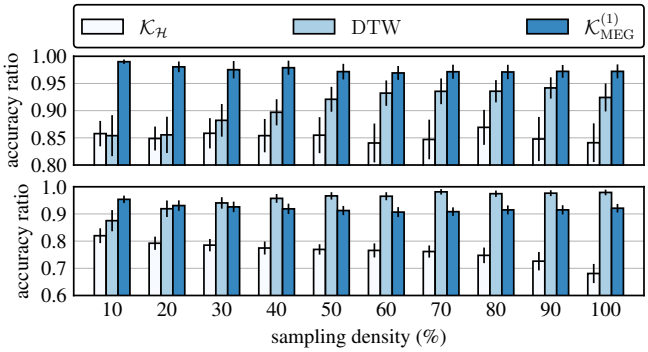


Figure 4: Comparison with other time series classification methods. The plot on the top corresponds to the 20 data sets whose optimal warping window size is at most 12; the plot on the bottom corresponds to the rest of 23 data sets with optimal warping window size greater than 12.

kernel $\mathcal{K}_{\mathcal{H}}$ proposed by Lu et al. [2008] is a time series kernel also designed for irregularly sampled time series. The RKHS kernel is defined as the squared norm between two posterior GP mean functions in the RKHS induced by a common prior covariance function. The kernel can be computed in closed form, but also discards posterior uncertainty since it only depends on the posterior GP means. It is also not possible to focus the kernel on assessing similarity over a specific time interval.

Dynamic time warping [Berndt and Clifford, 1994; Sakoe and Chiba, 1971] (DTW) is a widely used distance function for misaligned and warped time series. We compare to the 1-nearest neighbor algorithm using DTW distance subject to the Sakoe-Chiba warping constraint using the optimal window size published along with the UCR time series archive. Since classic DTW is not designed for irregularly sampled time series data, we also use GP regression to interpolate each time series on the same set of reference time points as the MSM kernels, and use the posterior means as the input to DTW.

In this experiment, we split the data sets into two groups according to their published optimal warping window sizes. Smaller warping window size implies the corresponding time series are almost aligned and have minimal warping. The 23 data sets with optimal window size greater than 12 are selected as the *warped* group, which implies that the corresponding time series require significant alignment or warping before direct comparison. The need for alignment and warping violates the assumptions of the MSM kernel as well as the RKHS kernel, which does not explicitly take warping or alignment into account. The rest of the 20 data sets are regarded as the *aligned* group.

Figure 4 shows that the RKHS kernel $\mathcal{K}_{\mathcal{H}}$ consistently performs the worst on both groups, because it fails to focus on a finite time interval of interest where data points are observed and does not account for uncertainty. For

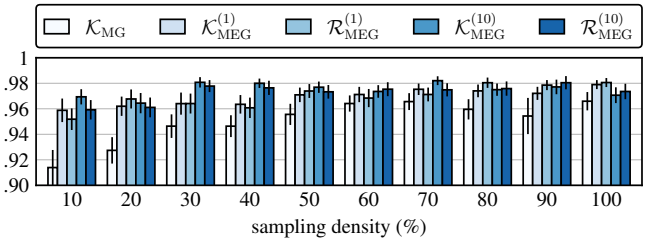


Figure 5: Comparison of the classification accuracy ratio of the exact time series expected Gaussian kernel against the random Fourier feature approximation. The baseline method $\mathcal{K}_{\text{MG}}$ is included as a reference.

the aligned group, our method always outperforms DTW. When the time series is more sparsely sampled, the advantage of our uncertainty-aware framework becomes more significant. For the warped group, DTW achieves better classification accuracy under low sparsity, but our approach achieves comparable or better accuracy under high sparsity while the RKHS kernel does not.

Random Features for MEG Kernels. To evaluate the random feature approximation to the MEG kernel in terms of classification accuracy, we use $m = \lfloor 10{,}000/k \rfloor$ random features to approximate the expected Gaussian kernel on each of k marginals, so that the total number of random features $M = mk$ is at most 10,000. In the experiment, $\mathcal{R}^{(1)}_{\text{MEG}}$ and $\mathcal{R}^{(10)}_{\text{MEG}}$ denote the random feature approximation for $\mathcal{K}^{(1)}_{\text{MEG}}$ and $\mathcal{K}^{(10)}_{\text{MEG}}$ with window size 1 and 10.

Figure 5 shows that the random feature approximation produces similar classification results compared to the exact kernels for both marginal window sizes. The baseline method $\mathcal{K}_{\text{MG}}$ is included to show that even when the accuracy declines due to approximation, it still outperforms the baseline method.

Table 1 shows the classification training and prediction time on the four largest data sets in the UCR archive. We divide the training and prediction task using either $\mathcal{K}^{(1)}_{\text{MEG}}$ or $\mathcal{R}^{(1)}_{\text{MEG}}$ into two steps: first, computing the kernel matrix for $\mathcal{K}^{(1)}_{\text{MEG}}$ or the random feature matrix for $\mathcal{R}^{(1)}_{\text{MEG}}$, which are shown as the 3rd and 5th column (denoted prep.) in Table 1; second, training and prediction time spent solely in the classifier, denoted train and test in the table.

The results in Table 1 show that computing the feature/kernel matrix dominates the entire training/prediction task. It is consistent with the time and space complexity analysis given in Table 2. In terms of the data size, computing the exact kernel takes $\mathcal{O}(n^2)$ time, while computing the random feature matrix takes $\mathcal{O}(n)$ time. As the window size d increases, computing the exact kernel takes $\mathcal{O}(d^3)$ time as oppose to $\mathcal{O}(d^2)$ for random feature approximation.

As for the actual classifier learning and prediction time, we can see that $\mathcal{R}^{(1)}_{\text{MEG}}$ takes longer than $\mathcal{K}^{(1)}_{\text{MEG}}$. This is because

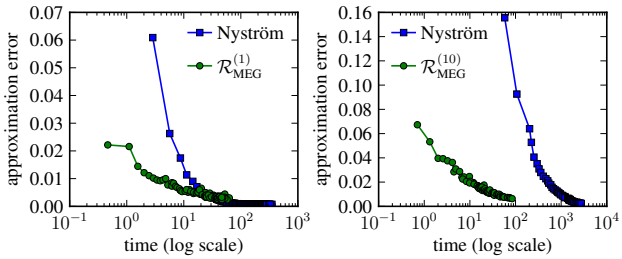
Table 1: Comparison of classification time (in seconds) on the four largest data sets using exact expected Gaussian kernels as opposed to random feature approximation under window sizes 1 and 10. In the table, the MEG kernel subscripts are dropped from the notation for brevity. The two numbers (n, L) for each data set denote the number of examples and the number of reference time points. Note that a MEG kernel with window size d consists of $k = L - d + 1$ base kernels (see Section 3).

data	meth.	prep.	train	prep.	test
TwoLead. (1162, 246)	$\mathcal{K}^{(1)}$	13.88	0.01	13.76	0.00
	$\mathcal{R}^{(1)}$	0.91	1.44	0.37	0.01
	$\mathcal{K}^{(10)}$	754.05	0.01	743.41	0.00
	$\mathcal{R}^{(10)}$	10.73	1.91	5.32	0.01
yoga (3300, 500)	$\mathcal{K}^{(1)}$	143.11	0.25	144.67	0.01
	$\mathcal{R}^{(1)}$	2.30	33.56	1.09	0.02
	$\mathcal{K}^{(10)}$	10751.85	0.32	10620.35	0.01
	$\mathcal{R}^{(10)}$	50.44	31.08	12.86	0.02
wafer (7164, 456)	$\mathcal{K}^{(1)}$	650.98	0.15	652.41	0.03
	$\mathcal{R}^{(1)}$	4.88	8.65	2.27	0.06
	$\mathcal{K}^{(10)}$	50452.40	0.17	49159.89	0.05
	$\mathcal{R}^{(10)}$	58.76	45.13	29.71	0.09
StarLight. (9236, 500)	$\mathcal{K}^{(1)}$	1103.91	0.21	1119.40	0.05
	$\mathcal{R}^{(1)}$	5.97	55.42	2.63	0.12
	$\mathcal{K}^{(10)}$	86676.10	0.46	83357.72	0.15
	$\mathcal{R}^{(10)}$	99.30	17.44	35.26	0.11

the final kernel matrix for $\mathcal{K}^{(1)}_{\text{MEG}}$ can be stored in $\mathcal{O}(n^2)$ space, as oppose to $\mathcal{O}(nmk)$ for $\mathcal{R}^{(1)}_{\text{MEG}}$, which is notably larger for our choice of m ($mk \approx 10{,}000$). By adjusting m, the total time using $\mathcal{R}^{(1)}_{\text{MEG}}$ can be further reduced, but it is already significantly faster overall with the value of m used here.

Comparing Random Features to Nyström Method. The Nyström method [Williams and Seeger, 2001] is a commonly used kernel approximation algorithm based on low-rank approximation to the full kernel matrix computed using a subset of the training data. We compare the time versus kernel approximation error trade-off when approximating the MEG kernel by the Nyström method and random features using window sizes 1 and 10. The experiment is conducted on the largest data set in the UCR archive, StarLightCurves, at 10% sampling density. For Nyström method, we plot the results using $s = 10, 20, \ldots, 500$ samples. For the random feature approximation, we plot the results using $m = 1, 2, \ldots, 100$ random features for each expected Gaussian kernel (at most $50{,}000$ total features for the largest m). Note that the number of training cases used for the Nyström method is at most the size of the training data; however, the number of random features can exceed the size of the training data.

The results show that the Nyström approximation can achieve higher approximation accuracy than random features when sufficient training data samples are used [Yang



(a) $d = 1, n = 9236$ (b) $d = 10, n = 1000$

Figure 6: Comparing time versus approximation error (in terms of the relative error $\mathbb{E}\|\widehat{\mathbf{K}} - \mathbf{K}\|/\|\mathbf{K}\|$) of Nyström method under various sizes of the subset of training data and random feature approximation with various numbers of random features.

Table 2: Comparing time and space complexity of the classification using the MEG kernel with window size d constructed from k expected Gaussian kernels, where n is the training data size, n' is the test data size. Nyström method uses a subset of the training data of size s, and $\mathcal{R}^{(d)}_{\text{MEG}}$ uses a total of $M = mk$ random features.

	$\mathcal{K}^{(d)}_{\text{MEG}}$	Nyström	$\mathcal{R}^{(d)}_{\text{MEG}}$
train time	$\mathcal{O}(d^3n^2k)$	$\mathcal{O}(d^3nsk + s^3)$	$\mathcal{O}(d^2nM)$
test time	$\mathcal{O}(d^3n'nk)$	$\mathcal{O}(d^3n'sk + n's^2)$	$\mathcal{O}(d^2n'M)$
train space	$\mathcal{O}(n^2)$	$\mathcal{O}(ns)$	$\mathcal{O}(nM)$
test space	$\mathcal{O}(n'n)$	$\mathcal{O}(n's)$	$\mathcal{O}(n'M)$

et al., 2012]. However, for d-dimensional Gaussians (for the MEG kernel with window size d), computing a single entry of the expected Gaussian kernel takes $\mathcal{O}(d^3)$ comparing to $\mathcal{O}(d^2)$ to compute a single random feature, as in the case of comparing to exact kernel computation. The detailed complexity analysis is given in Table 2. Figure 6(b) shows that for window size 10, the random feature approximation needs significantly less time to achieve an acceptable error rate.

Fastfood Method for High-Dimensional Marginals. We compare the straightforward random feature computation to two acceleration methods using Fastfood as described in Section 4.2. This experiment is conducted on the data set StarLightCurves from the UCR archive at 10% sampling density with the full window size $L = 500$. That is, there is a single expected Gaussian kernel with $d = 500$ in the MEG kernel. We use randomized truncated SVD [Halko et al., 2011] for the low-rank approximation of covariance matrices with rank $r = 10$.

Figure 7 shows the time-versus-error relationship using three different methods with 2^ℓ random features for $\ell = 9, \ldots, 13$ (from top to bottom). It shows that all three methods achieve similar errors (relative error in terms of spectral norms) when using the same feature size. However, the Fastfood method using (5), denoted Fastfood in the figure, is at least 14 times faster among five feature sizes than the

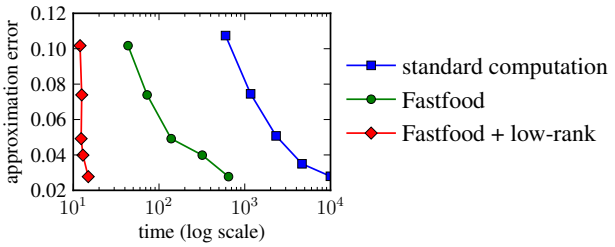


Figure 7: Comparing time versus approximation error (relative error) of the standard feature computation and the two Fastfood methods under $d = 500$ and $n = 1000$. The five points for each method correspond to using 512, 1024, 2048, 4096, and 8192 features from top to bottom.

standard method, due to the $\mathcal{O}(nMd \log d)$ time for Fastfood as opposed to $\mathcal{O}(nMd^2)$. With low-rank covariance approximation, the running time can be improved significantly again, even if an extra truncated SVD is required. The SVD overhead is roughly a constant of 2.7 seconds, which accounts for 86% time in the smallest case (512 features) and 42% in the largest case (8192 features).

6 CONCLUSIONS AND FUTURE WORK

We have proposed a kernel-based framework for classification of sparse and irregularly sampled time series that re-represents time series using Gaussian process and then assesses the similarity between the GPs based on the similarity between their finite marginals defined over sliding time windows. Our results show that the proposed approach achieves better average accuracy on a large time series classification benchmark compared to all other methods considered when the time series are alignedor under high sparsity. Further, our extension to random Fourier features achieves significant speedups relative to exact kernel computations as well as Nyström approximation on large time series data sets. Our application of Fastfood and low-rank covariance approximations yields further speedups in the case where large-dimensional marginals are required.

Possible directions for future work include learning kernel combination weights, extending the MSM framework to multi-output Gaussian processes for multivariate time series, and the extension of this framework to distributions other than Gaussians or different base kernels. Moreover, instead of performing classification on the random features using linear SVMs, we can use the random feature vector as an uncertainty-aware embedding of the data in various deep learning architectures, as well as unsupervised learning models for problems like clustering.

Acknowledgements

This material is based upon work supported by the National Science Foundation under Grant No. 1350522.

References

Bach, F. R., Lanckriet, G. R., and Jordan, M. I. (2004). Multiple kernel learning, conic duality, and the smo algorithm. In *Proceedings of the twenty-first international conference on Machine learning*, page 6. ACM.

Berndt, D. J. and Clifford, J. (1994). Using dynamic time warping to find patterns in time series. In *KDD workshop*. Seattle, WA.

Chang, C.-C. and Lin, C.-J. (2011). LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

Clark, J. and Bjørnstad, O. (2004). Population time series: process variability, observation errors, missing values, lags, and hidden states. *Ecology*, 85(11):3140–3150.

Cortes, C. and Vapnik, V. (1995). Support-vector networks. *Machine learning*, 20(3):273–297.

Fan, R.-E., Chang, K.-W., Hsieh, C.-J., Wang, X.-R., and Lin, C.-J. (2008). LIBLINEAR: A library for large linear classification. *Journal of Machine Learning Research*, 9:1871–1874.

Halko, N., Martinsson, P.-G., and Tropp, J. A. (2011). Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. *SIAM review*, 53(2):217–288.

Hensman, J., Fusi, N., and Lawrence, N. D. (2013). Gaussian processes for big data. In *Proceedings of the Twenty-Ninth Conference on Uncertainty in Artificial Intelligence, Bellevue, WA, USA, August 11-15, 2013*.

Jebara, T., Kondor, R., and Howard, A. (2004). Probability product kernels. *The Journal of Machine Learning Research*, 5:819–844.

Keogh, E., Xi, X., Wei, L., and Ratanamahatana, C. A. (2011). The UCR time series classification/clustering homepage: `www.cs.ucr.edu/~eamonn/time_series_data/`.

Le, Q., Sarlós, T., and Smola, A. (2013). Fastfood–approximating kernel expansions in loglinear time. In *ICML*.

Lopez-Paz, D., Sra, S., Smola, A. J., Ghahramani, Z., and Schölkopf, B. (2014). Randomized nonlinear component analysis. In *ICML*.

Lu, Z., Leen, T. K., Huang, Y., and Erdogmus, D. (2008). A reproducing kernel hilbert space framework for pairwise time series distances. In *Proceedings of the 25th international conference on Machine learning*, pages 624–631. ACM.

Marlin, B. M., Kale, D. C., Khemani, R. G., and Wetzel, R. C. (2012). Unsupervised pattern discovery in electronic health care data using probabilistic clustering models. In *Proceedings of the 2nd ACM SIGHIT International Health Informatics Symposium*, pages 389–398.

Quiñonero-Candela, J. and Rasmussen, C. E. (2005). A unifying view of sparse approximate gaussian process regression. *The Journal of Machine Learning Research*, 6:1939–1959.

Rahimi, A. and Recht, B. (2007). Random features for large-scale kernel machines. In *Advances in neural information processing systems*, pages 1177–1184.

Rasmussen, C. and Williams, C. (2006). *Gaussian processes for machine learning*.

Ruf, T. (1999). The lomb-scargle periodogram in biological rhythm research: analysis of incomplete and unequally spaced time-series. *Biological Rhythm Research*, 30(2):178–201.

Sakoe, H. and Chiba, S. (1971). A dynamic programming approach to continuous speech recognition. In *Proceedings of the Seventh International Congress on Acoustics*, volume 3, pages 65–69.

Scargle, J. D. (1982). Studies in astronomical time series analysis. ii-statistical aspects of spectral analysis of unevenly spaced data. *The Astrophysical Journal*, 263:835–853.

Schulz, M. and Stattegger, K. (1997). Spectrum: Spectral analysis of unevenly spaced paleoclimatic time series. *Computers & Geosciences*, 23(9):929–945.

Smola, A., Gretton, A., Song, L., and Schölkopf, B. (2007). A hilbert space embedding for distributions. In *Algorithmic Learning Theory*, pages 13–31. Springer.

Tropp, J. A. (2012a). User-friendly tail bounds for sums of random matrices. *Foundations of Computational Mathematics*, 12(4):389–434.

Tropp, J. A. (2012b). User-friendly tools for random matrices: An introduction. Technical report, DTIC Document.

Williams, C. and Seeger, M. (2001). Using the Nyström method to speed up kernel machines. In *Proceedings of the 14th Annual Conference on Neural Information Processing Systems*.

Xing, E. P., Jordan, M. I., Russell, S., and Ng, A. Y. (2002). Distance metric learning with application to clustering with side-information. In *Advances in neural information processing systems*, pages 505–512.

Yang, T., Li, Y.-F., Mahdavi, M., Jin, R., and Zhou, Z.-H. (2012). Nyström method vs random fourier features: A theoretical and empirical comparison. In *Advances in neural information processing systems*, pages 476–484.