# TIME SIGNAL CLASSIFICATION USING RANDOM CONVOLUTIONAL FEATURES

*Abelino Jiménez & Bhiksha Raj*

Carnegie Mellon University
Email: abjimenez@cmu.edu, bhiksha@cs.cmu.edu

**ABSTRACT**

In this paper we present a transformation to convert time signals into a randomized low-dimensional vectors such that the inner product between these new features provides information about the similarity of the signals. We show that the described inner product approximates a cross-correlation based kernel. This is very useful at the moment of use Kernel Machines, such as Non-linear Support Vector Machines. Indeed, this allows to apply simpler and faster linear methods on the generated random features. Our proposed scheme improves computational storage and time cost over the direct kernel approach, while performing the classification performance with minimal loss. We support our statements by providing theoretical guarantees as well as empirical evaluation across different data sets.

***Index Terms—*** Time Signal Classification, Random Convolutional Features, Kernel Machines.

## 1. INTRODUCTION

One particularly important task in time signal processing is signal classification, where given a time signal, we must determine the class it belongs to. Applications range from classification of medical time signals (such as ECGs) to automatic speech recognition, classification of financial series, astronomy, and even modeling language.

The most common approaches employ statistical models, such as Hidden Markov Models [1] [2] [3], conditional random fields [4] [5], or convolutional or recurrent neural networks [6] [7] [8] [9]. While highly effective, these approaches are typically data intensive, prone to overfitting, and often need high tuning of hyperparameters.

An alternate approach is through *kernel* methods, such as support vector machines, which are generally less data intensive and easier to optimize. However, these come with a concomitant challenge: that of defining an appropriate kernel that captures the similarity between instances. Conventional similarity measures such as the inner product will not suffice, since one must also consider the *alignment* between time signals when computing their similarity. Consider, for instance, the two signals $sin(\omega k)$ and $sin(\omega k + \phi)$, where $k$ is the running time index. Although the two are structurally identical, the inner product between the two will vary with the phase shift $\phi$. In order to correctly compute the similarity between the two signals, they must be *aligned* to line up; however the amount of shift will generally not be known *a priori*. Hence, a good similiarity measure between time signals must be *shift invariant*.

Several common solutions for computing kernels between time signals view them as regular vectors and do not consider the alignment between them [10]. Other approaches do implicitly consider the issue, but generally construct expensive statistical machinery to do so [11][12] [13] [14]. Fourier-transform-based methods transform the signals into the Fourier domain, to deal with the issue of shift invariance [15], however they effectively consider every alignment, not just the right one, making them susceptible to noise [16]. The kernels that best account for the alignment between the two signals are DTW-based kernels [17] [18] that explicitly find the optimal warping between them in the process of computing their similarity. However, the computational complexity of finding the optimal warp is quadratic, or even cubic in the length of the sequences, making them impractical in many scenarios.

Possibly the most appropriate similarity measure in this setting is the *peak cross-correlation* between the signals. The cross-correlation of two time signals computes the inner product between the signals at *every* alignment between the two. Formally, given two time signals $\mathbf{f} = (f_1, ..., f_N)$ and $\mathbf{g} = (g_1, ..., g_N)$ of length $N$, the cross-correlation between the two, represented as $\mathbf{f} \star \mathbf{g}$, is the signal computed as

$$(\mathbf{f} \star \mathbf{g})_i = \sum_k f_k \cdot g_{k+i} \tag{1}$$

Different settings of the limits of the summation in $k$ result in different types of cross-correlation. The *linear* cross-correlation between $\mathbf{f}$ and $\mathbf{g}$ is obtained by summing $k$ from $-\infty$ to $\infty$, assuming that both $\mathbf{f}$ and $\mathbf{g}$ take value 0 outside the index range $1 \cdots N$. The *circular* cross-correlation is obtained by summing $k$ from 1 to $N$, and assuming $g_{k+i} = g_{N-k-i}$ for $k + i > N$ [19]. The linear cross-correlation may also be obtained from the circular cross-correlation by zero-padding both signals out to length $2N$ and computing the circular cross-correlation over the longer zero-padded sequences. In either case, the outcome of Equation 1 is a complete time signal, where each term $(\mathbf{f} \star \mathbf{g})_i$ computes the inner product between $\mathbf{f}$ and $\mathbf{g}$ when the latter has been shifted by $i$ in order to line it up with $\mathbf{f}$.

The *peak* cross correlation is the maximal value of the cross correlation, $\max_i(\mathbf{f} \star \mathbf{g})_i$, which is the inner product under the best scoring alignment between the two time signals. Although this quantity has good properties such as symmetry, and accounts well for the alignment of the two signals, it does not define a valid *kernel*; the corresponding gram matrix may not be positive definite [20]. Instead, we define a *cross-correlation kernel* as a weighted combination of the elements of the cross-correlation, such that the maximum value of the cross-correlation has a larger weight compared to the remaining values:

$$\mathcal{K}(\mathbf{f}\,,\,\mathbf{g}) = \frac{1}{N} \sum_i \exp\left(\gamma \cdot (\mathbf{f} \star \mathbf{g})_i\right), \tag{2}$$

where $\gamma$ is a positive parameter which governs the relative contribution of the individual terms – as $\gamma$ increases, the contribution of the maximum values of the cross-correlation is larger. It can be proved that $\mathcal{K}(\mathbf{f}\,,\,\mathbf{g})$ as defined above is indeed a proper kernel [20].

For the *circular* cross-correlation, one way to compute this is through the Fourier Transform. Using the properties of the Fourier Transform it is straightforward to see that [19]

$$\mathcal{K}(\mathbf{f}\,,\,\mathbf{g}) = \frac{1}{N} \sum_i \exp\left(\gamma \cdot \mathcal{F}^{-1}\left(\mathcal{F}(\mathbf{f}) \cdot \mathcal{F}(\mathbf{g})^*\right)_i\right) \tag{3}$$

where $\mathcal{F}(\mathbf{f})$ is the Fourier transform of $\mathbf{f}$, $\mathcal{F}^{-1}$ is the inverse Fourier transform operator, and $\mathcal{F}(\mathbf{g})^*$ is the component-wise complex conjugate of $\mathcal{F}(\mathbf{g})$. Using the Fast Fourier Transform algorithm to compute the Fourier transforms, the complexity of computing $\mathcal{K}(\mathbf{f}\,,\,\mathbf{g})$ is $\mathcal{O}(N \log N)$ [21]. While this is less computationally expensive to compute than other time-signals kernels, it is still significantly more expensive than computing a simple inner product, which has a complexity of only $\mathcal{O}(N)$. The difference is even greater when the sequences must be increased in length by zero-padding, in order to compute the linear cross-correlation via circular cross-correlation.

This leads us to the key challenge addressed in this paper. The cost of computing the kernel is a significant component of both training and evaluating kernel machines such as SVMs. For SVMs in particular, for non-linear kernels such as the cross-correlation kernel, we must solve the dual form of the optimization to train the model, which requires the computation of a gram matrix. Gram matrix computation scales quadratically (both in time and storage complexity) with the number of training samples. Thus, for the cross-correlation kernel, training the SVM with $L$ training samples would require $\mathcal{O}(L^2 N \log N)$ computations, just for the gram matrix. In addition, given a model with $V$ support vectors, the inference would require $\mathcal{O}(V N \log N)$ computations per sample to be classified.

In such situations, it is well known that *random projections* of the data can be used to derive simpler, lower-dimensional representations of the data on which the kernels can be approximated, to derive savings in both computation and storage [22] [23]. This method has been successfully applied to both inner-product kernels of high-dimensional data [24] [25], and to a variety of non-linear kernels [26] [27], although in the latter case the actual manner of computing the random features depends on the kernel.

In this paper we extend this principle to derive a randomized scheme for computing lower-dimensional representations of time-signals data, from which cross-correlation kernels can be computed through a simple inner-product computation. We call these *random convolutional features*.

The proposed method has multiple desirable features. The computation of random features makes no assumption about the length of the time signals; this is, the number of components in the low-dimensional representation depends only on the *energy* in the signal, and not on the number of samples in it. In practice, the number of components required is generally smaller than the length of the time signal itself. Thus the cost of computing a kernel thus becomes that of computing a low-dimensional inner product. For energy-normalized signals, this becomes a *fixed*-dimensional computation. The final benefit is one that is common to most random-feature-based approximations of kernel functions: since kernel computation now becomes an inner product, the training of the SVM can be solved in its primal form, eliminating the need for the gram matrix. The time and space complexity of the model too simply become the size (number of components) in the random feature itself; the cost of inference reduces to that of deriving the projection itself.

This paper is organized as follows. In Section 2 we describe our proposed random feature construction providing some theoretical guarantees to validate its use. In Section 3, we present a experimental results on a number of data sets, showing that these benefits come at no cost to classification performance, which remains comparable to that obtained with the full cross-correlation kernel. Finally, in Section 4, we conclude analyzing the scope of this techniques and discussing future directions.

## 2. METHODS

In this section we describe the construction of our random feature scheme that approximates the cross-correlation kernel.

First, we present our scheme, named *random convolutional features*, which defines a random mapping from time signals to a low dimensional space, approximating the cross-correlation kernel through the computation of inner products between random features. Later, we show theoretical guarantees to validate the approximation to the cross-correlation kernel.

**Notation:** We use $N$ as the dimension of original vectors as well as the length of time series, $M$ as the dimension of random features, and $L$ as the number of training samples. We use uppercase letters to denote matrices and lowercase boldface letters to denote vectors. All norms in this paper are $\ell_2$-norm, denoted by $\|\cdot\|$.

### 2.1. Random Convolutional Features

As its name says, the random feature scheme presented in this paper is based on a convolutional operation. The randomness of this features comes from its parameters which are randomly generated according to some particular distributions.

**Definition.** Let $\gamma$ be a positive number. Let $\mathcal{W} = \{\mathbf{w}_1, ..., \mathbf{w}_M\}$ be a set of $M$ random time signals of length $N$, where each component is random and independently generated according to a Gaussian distribution $\mathcal{N}(0, \gamma)$, and $\mathcal{U}$ a set $M$ scalars where each one is random and independently generated using a uniform distribution between 0 and $2\pi$. We define a *random convolutional feature* as the random mapping $\psi_{\mathcal{W},\mathcal{U},\gamma} : \mathbb{R}^N \to \mathbb{R}^M$ as follows:

$$\psi_{\mathcal{W},\mathcal{U},\gamma}(\mathbf{f})_i \;=\; \frac{\sqrt{2}}{N\sqrt{M}} \exp\left(\frac{\gamma}{2}\|\mathbf{f}\|^2\right) \sum_{j=1}^{N} \cos\left((\mathbf{w}_i * \mathbf{f})_j + u_i\right)$$

where $*$ denotes the convolution operation between signals. Algorithm 1 summarizes the process to compute random features from time signals.

**Data:** Time signals $\{\mathbf{f}_1, \mathbf{f}_2, ..., \mathbf{f}_L\}$ with length $N$
**Result:** $\{\mathbf{z}_1, \mathbf{z}_2, ..., \mathbf{z}_L\}$ Random Convolutional Features
initialization;
**for** $i = 1, ..., M$ **do**
  Draw $\mathbf{w}_i$ random signal, with independent component and each as Gaussian with 0 mean and variance $\gamma$;
  Draw $u_i$ random scalar as uniform between 0 and $2\pi$;
  **for** $k = 1, ..., L$ **do**
    Compute $E = \frac{\sqrt{2}}{N\sqrt{M}} \exp\left(\frac{\gamma}{2}\|\mathbf{f}_k\|^2\right)$;
    Compute signal $\mathbf{s} = \mathbf{w}_i * \mathbf{f}_k$;
    Compute $C = \sum_j \cos(\mathbf{s}_j + u_i)$;
    Define $(\mathbf{z}_k)_i = E \cdot C$ ;
  **end**
**end**
**Return** $\{\mathbf{z}_1, \mathbf{z}_2, ..., \mathbf{z}_L\}$, where $\psi_{\mathcal{W},\mathcal{U},\gamma}(\mathbf{f}_k) = \mathbf{z}_k$

**Algorithm 1:** Random Convolutional Features computation

### 2.2. Theoretical Results

The following results show the relation of the random convolutional features to the cross-correlation kernel. The first result analyzes the expected values of inner products between random features while the second shows how to use these inner products.

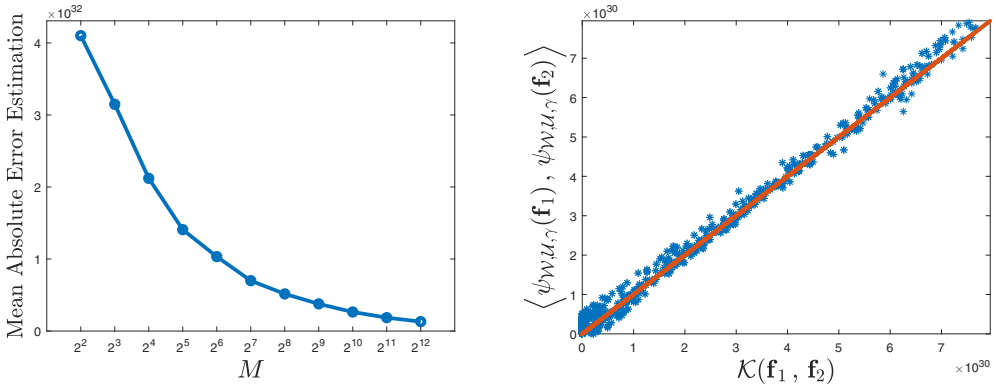


**Fig. 1**. Approximation of Cross-correlation Kernel through inner product of Random Features. (Left) Effect of the number of components in the random mapping on the approximation error for Kernel estimation. (Right) Pairwise comparison between the cross-correlation and its estimation through random features. Each dot is a corresponds to a pair $(\mathbf{f}_1, \mathbf{f}_2)$. We consider $M = 1024$. Both figures where generated using the Gun-Point data set from [28]

**Theorem 1.** For any time signals $\mathbf{f}_1,\ \mathbf{f}_2 \in \mathbb{R}^N$, the expected value of the inner product between their corresponding random mappings $\psi_{\mathcal{W},\mathcal{U},\gamma}(\mathbf{f}_1)$ and $\psi_{\mathcal{W},\mathcal{U},\gamma}(\mathbf{f}_2)$ is given by:

$$\mathbb{E}\Big(\big\langle \psi_{\mathcal{W},\mathcal{U},\gamma}(\mathbf{f}_1)\ ,\ \psi_{\mathcal{W},\mathcal{U},\gamma}(\mathbf{f}_2)\big\rangle\Big) \quad = \quad \mathcal{K}\big(\mathbf{f}_1\ ,\ \mathbf{f}_2\big)$$

$$= \quad \frac{1}{N}\sum_{i=1}^{N} \exp\left(\gamma \cdot (\mathbf{f}_1 \star \mathbf{f}_2)_i\right)$$

*Proof. (see appendix)*

Thus, the cross-correlation kernel corresponds to the expected value of the inner product between the proposed random features. Furthermore, we can analyze the convergence of actual inner product to the kernel function. The following statement, based on applying Hoeffding's inequality, provides some guarantees.

**Theorem 2.** If we have a set of $L$ time signals of length $N$, $\mathcal{T} = \{\mathbf{f}_1, \mathbf{f}_2, ..., \mathbf{f}_L\}$, and $C \geq \|\mathbf{f}_i\|^2$ for all $i$, then, considering $M \geq \frac{16\exp(2\gamma C)}{\varepsilon^2}\log\left(\frac{L}{\eta}\right)$, we have

$$\mathbb{P}\Big(\forall \mathbf{f}_i, \mathbf{f}_j \in \mathcal{T}\ ,\ \Big|\big\langle \psi_{\mathcal{W},\mathcal{U},\gamma}(\mathbf{f}_i)\ ,\ \psi_{\mathcal{W},\mathcal{U},\gamma}(\mathbf{f}_j)\big\rangle - \mathcal{K}\big(\mathbf{f}_i, \mathbf{f}_j\big)\Big| < \varepsilon\Big) \geq 1-\eta$$

Therefore, the probability of obtaining a small error in the estimation of the kernel using the inner product between random features depends on the number of components in the random features and the energy of the time signals. Figure 1 illustrates this.

Applying this technique to SVMs has several advantages. In case of using directly the cross-correlation kernel, we know that for training we need to obtain the gram matrix which requires $\mathcal{O}(L^2\, N \log N)$ computations. Alternatively, we can compute random features and train a linear model using a fast implementation. Obtaining the random features requires $\mathcal{O}(MLN \log N)$. So, if $M \ll L$, we can observe a natural advantage in the training phase. In the case of inference, using the kernel function directly requires to store $\mathcal{O}(VN)$ values and perform $\mathcal{O}(VN \log N)$ operations for each prediction, where $V$ is the number of support vectors. On the other hand, using random features, we must store $\mathcal{O}(M)$ values and perform $\mathcal{O}(MN \log N)$ computations for each prediction. Then, in case of $M \ll V$ we get a benefit for inference computation, while having $M \ll VN$ gives benefits in model storage.

## 3. EXPERIMENTS

As a proof of concept, we applied the presented technique over 15 different data sets for the task of time signals binary classification. This section aims to show the benefits of the modeling capabilities of the cross-correlation kernel as well as analyze the benefits of using random convolutional features with linear SVM over the raw time signals with the cross-correlation kernel.

### 3.1. Datasets

For our experiments, we used 15 data sets provided by the University of California, Riverside time series classification archive [28]. Each data set consists of equal-length time series belonging to one of two classes. The data sets selected and details about the size of the training and testing sets, as well as the time series length are presented in table 1.

Beside the sample points, this archive also provides the prediction error of three different classification methods; 1-Nearest Neighbor using Euclidean distance, 1-Nearest Neighbor using Best Warping Window DTW, and 1-Nearest Neighbor using DTW with no Warping window.

### 3.2. Results using Nonlinear SVM with Cross-correlation kernel

First, we evaluate the capabilities of the cross-correlation kernel over the 15 data sets presented in table 1 using a SVM classifier based on this kernel.

To select the hyperparameters $\gamma$ that defines the kernel, and $C$ corresponding to the SVM cost, we performed a grid search considering $\gamma \in \{2^{-10}, 2^{-9}, ..., 2^{-1}, 1\}$ and $C \in \{10^{-4},\ 10^{-2}, 1\ , 10^{2}\ ,\ 10^{4}\}$, doing the selection through jack-knife cross-validation. In table 1 the selected parameters can be found, as well as their corresponding error classification rate.

We can observe that in 11 of 15 data sets, SVM models with cross-correlation kernel outperform alternative methods. This shows empirically the utility of this kernel function.

**Table 1**. Results in Binary Classification Task.

| Data set | | SVM with Cross-Correlation Kernel | | | SVM with Random Convolutional Features | | | | Other Methods | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | Signal Length | $\gamma$ | $C$ | Error Rate | $M = 2^5$ | $M = 2^6$ | $M = 2^7$ | $M = 2^8$ | 1-NN Euclidean Distance | 1-NN Best Warping Window DTW | 1-NN DTW, no Warping Window |
| Gun-Point | 150 | 0.0625 | 0.0001 | **0.013** | 0.035 | 0.031 | 0.028 | 0.025 | 0.087 | 0.087 | 0.093 |
| Lightning-2 | 637 | 0.0156 | 0.0001 | 0.230 | 0.282 | 0.252 | **0.224** | 0.212 | 0.246 | 0.131 | 0.131 |
| ECG | 96 | 0.0313 | 1 | **0.100** | 0.159 | 0.142 | 0.130 | 0.129 | 0.120 | 0.120 | 0.230 |
| Coffee | 286 | 0.0313 | 0.0001 | 0.000 | 0.012 | **0.007** | **0.003** | **0.000** | 0.000 | 0.000 | 0.000 |
| ECGFiveDays | 136 | 0.1250 | 0.0001 | **0.005** | 0.009 | 0.003 | 0.006 | **0.002** | 0.203 | 0.203 | 0.232 |
| MoteStrain | 84 | 0.2500 | 0.0001 | 0.265 | 0.274 | 0.246 | 0.238 | **0.255** | 0.121 | 0.134 | 0.165 |
| SonyAIBORobot Surface | 70 | 0.1250 | 0.0001 | **0.068** | 0.237 | 0.212 | 0.183 | 0.182 | 0.305 | 0.305 | 0.275 |
| SonyAIBORobot SurfaceII | 65 | 0.0625 | 0.0100 | **0.132** | 0.199 | 0.194 | 0.187 | 0.168 | 0.141 | 0.141 | 0.169 |
| TwoLeadECG | 82 | 0.2500 | 0.0001 | **0.011** | 0.036 | 0.024 | 0.021 | **0.016** | 0.253 | 0.132 | 0.096 |
| BeetleFly | 512 | 0.0156 | 0.0001 | **0.050** | 0.303 | 0.292 | 0.270 | 0.242 | 0.250 | 0.300 | 0.300 |
| BirdChicken | 512 | 0.0078 | 0.0100 | **0.200** | 0.171 | 0.176 | 0.185 | **0.204** | 0.450 | 0.300 | 0.250 |
| Ham | 431 | 0.0313 | 0.0001 | **0.305** | **0.309** | 0.311 | **0.309** | **0.300** | 0.400 | 0.400 | 0.533 |
| Herring | 512 | 0.0010 | 100,000 | **0.359** | 0.418 | 0.424 | 0.421 | 0.407 | 0.484 | 0.469 | 0.469 |
| ToeSegmentation1 | 277 | 0.0313 | 0.0001 | **0.118** | 0.274 | 0.248 | 0.225 | 0.221 | 0.320 | 0.250 | 0.228 |
| ToeSegmentation2 | 343 | 0.0039 | 1.0000 | 0.123 | 0.182 | 0.157 | 0.140 | 0.136 | 0.192 | 0.092 | 0.162 |

### 3.3. Results using Linear SVM with Random Features

To analyze the utility of our proposed random features we trained linear SVMs using different values of $M$. To select the hyperparameters we proceeded as before. In table 1 we present the error rates we obtained. We can observe how the error rate changes as the number of components on the random features increases. In general, we observe that error rate decreases as $M$ increases; except for the case of the data set *BirdChicken*.

Moreover, in 7 of 15 data sets the difference between the non-linear method compared to the linear method is less than 1% using 256 or less components in random features, and in other 4 data sets the difference is less than 5%. This diversity is due the complexity of finding the decision boundary, as is established in [24]. We note that, even with 32 components the loss of performance be can considered acceptable in several cases.

## 4. CONCLUSIONS

In this paper we presented a random feature scheme that allows us to transform time signals into a low dimensional vectors, and used them to approximate the cross-correlation kernel through a simple inner product computation. Our main contribution was to show theoretical guarantees to validate this approximation. We studied this scheme with SVMs for Time Signal Classification. The proposed random feature provides minimal loss of performance in several cases, reducing storage and transforming a non-linear learning problem into a linear one. This has a significant implication in big data scenarios, where a large number of signals must be processed.

Moreover, the presented scheme has other potential applications. For example, with this technique we could train compact models on devices; linear models are much simpler to train than kernel-based models, specially with low dimensional data. Another application is related to cloud computing and privacy. In case we need to process sensitive time signals (e.g. medical or financial data), we can apply this random mapping keeping the random parameters $\mathcal{W}$ and $\mathcal{U}$ private. Then, we can still process the transformed data in the cloud, hiding in somehow information without exposing the original data.

Finally, we think this work can provide some guidelines to understand other methods based on cross-correlation, such as Convolutional Neural Networks. In fact, previous works have shown that it is possible to obtain good performance even without training the random initialized filters [29], being consistent with our results.

## 5. APPENDIX

*Proof Theorem 1.*
Since $\mathbf{w}$ are totally random, we can interchange the convolutional operation by a cross-correlation.

$$\begin{aligned}
&\mathbb{E}\left(\psi_{\mathcal{W},\mathcal{U},\gamma}(\mathbf{f}_1)_i \cdot \psi_{\mathcal{W},\mathcal{U},\gamma}(\mathbf{f}_2)_i\right) \\
&\quad = \frac{2}{N^2 M}\exp\left(\frac{\gamma}{2}\|\mathbf{f}_1\|^2\right)\cdot\exp\left(\frac{\gamma}{2}\|\mathbf{f}_2\|^2\right)\cdot \\
&\mathbb{E}\left[\left(\sum_{j=1}^{N}\cos\left((\mathbf{w}_i \star \mathbf{f}_1)_j + u_i\right)\right)\cdot\left(\sum_{j=1}^{N}\cos\left((\mathbf{w}_i \star \mathbf{f}_2)_j + u\right)\right)\right] \\
&\quad = \frac{2}{N^2 M}\exp\left(\frac{\gamma}{2}\|\mathbf{f}_1\|^2\right)\cdot\exp\left(\frac{\gamma}{2}\|\mathbf{f}_2\|^2\right)\cdot \\
&\quad \mathbb{E}\left[\sum_{j=1,k=1}^{N}\cos\left((\mathbf{w}_i \star \mathbf{f}_1)_j + u_i\right)\cdot\cos\left((\mathbf{w}_i \star \mathbf{f}_2)_k + u_i\right)\right]
\end{aligned}$$

but, by definition of cross-validation, we know that $(\mathbf{w} \star \mathbf{f})_j = \langle \mathbf{w}\,,\, \mathbf{f}_{(j)}\rangle$, where $\mathbf{f}_{(j)}$ is the circular shifted version of $\mathbf{f}$ shifted by $j$ positions. Then, from [25], we have

$$\begin{aligned}
&\mathbb{E}\left[\psi_{\mathcal{W},\mathcal{U},\gamma}(\mathbf{f}_1)_i \cdot \psi_{\mathcal{W},\mathcal{U},\gamma}(\mathbf{f}_2)_i\right] \\
&\quad = \frac{2}{N^2 M}\exp\left(\frac{\gamma}{2}\|\mathbf{f}_1\|^2\right)\cdot\exp\left(\frac{\gamma}{2}\|\mathbf{f}_2\|^2\right)\cdot \\
&\sum_{j=1,k=1}^{N}\mathbb{E}\Big[\cos\left((\mathbf{w}_i \star \mathbf{f}_1)_j + u_i\right)\cdot\cos\left((\mathbf{w}_i \star \mathbf{f}_2)_k + u_i\right)\Big] \\
&\quad = \frac{1}{N^2 M}\sum_{j=1,k=1}^{N}\exp(\gamma\langle \mathbf{f}_{1(j)}\,,\, \mathbf{f}_{2(k)}\rangle)
\end{aligned}$$

but, we know that $\langle \mathbf{f}_{1(j)}\,,\, \mathbf{f}_{2(k)}\rangle = \langle \mathbf{f}_1\,,\, \mathbf{f}_{2(k-j)}\rangle$. Therefore,

$$\begin{aligned}
\mathbb{E}\left[\psi_{\mathcal{W},\mathcal{U},\gamma}(\mathbf{f}_1)_i \cdot \psi_{\mathcal{W},\mathcal{U},\gamma}(\mathbf{f}_2)_i\right] &= \frac{1}{N^2 M}\cdot N\sum_{j=1}^{N}\exp(\gamma\langle \mathbf{f}_1\,,\, \mathbf{f}_{2(j)}\rangle) \\
&= \frac{1}{NM}\sum_{j=1}^{N}\exp(\gamma(\mathbf{f}_1 \star \mathbf{f}_2)_j)
\end{aligned}$$

Finally, using the linearity of expectation, we get the desired result. □

## 6. REFERENCES


[1] Peter Sykacek and Stephen J. Roberts. *"Bayesian time series classification."* In Advances in Neural Information Processing Systems, pp. 937-944. 2002.

[2] Bilal Esmael, Arghad Arnaout, Rudolf K. Fruhwirth, and Gerhard Thonhauser. *"Improving time series classification using Hidden Markov Models."* In Hybrid Intelligent Systems (HIS), 2012 12th International Conference on, pp. 502-507. IEEE, 2012

[3] Karan Sikka, Abhinav Dhall, and Marian Bartlett. *"Exemplar hidden markov models for classification of facial expressions in videos."* In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, pp. 18-25. 2015

[4] Minyoung Kim. *"Semi-supervised learning of hidden conditional random fields for time-series classification."* Neurocomputing 119 (2013): 339-349.

[5] Myriam Abramson. *"Sequence classification with neural conditional random fields."* In Machine Learning and Applications (ICMLA), 2015 IEEE 14th International Conference on, pp. 799-804. IEEE, 2015.

[6] Bendong Zhao, Huanzhang Lu, Shangfeng Chen, Junliang Liu, and Dongya Wu. *"Convolutional neural networks for time series classification."* Journal of Systems Engineering and Electronics 28, no. 1 (2017): 162-169.

[7] Nima Hatami, Yann Gavet, and Johan Debayle. *"Classification of time-series images using deep convolutional neural networks."* In Tenth International Conference on Machine Vision (ICMV 2017), vol. 10696, p. 106960Y. International Society for Optics and Photonics, 2018.

[8] Fazle Karim, Somshubra Majumdar, Houshang Darabi, and Shun Chen. *"Lstm fully convolutional networks for time series classification."* IEEE Access 6 (2018): 1662-1669.

[9] Yang Guo, Zhenyu Wu, and Yang Ji. *"A Hybrid Deep Representation Learning Model for Time Series Classification and Prediction."* In Big Data Computing and Communications (BIGCOM), 2017 3rd International Conference on, pp. 226-231. IEEE, 2017.

[10] Stefan Ruping. *"Svm kernels for time series analysis"*. In Klinkenberg, R., Ruping, S., Fick, A., Henze, N., Herzog, C., Molitor, R., and Schroder, O., editors, LLWA 01 - Tagungsband der GI-Workshop-Woche Lernen - Lehren - Wissen - Adaptivitat, Forschungsberichte des Fachbereichs Informatik der Universitat Dortmund, pages 4350, Dortmund, Germany. 2001.

[11] Ginés Rubio, Héctor Pomares, Luis J. Herrera and Ignacio Rojas. *"Kernel Methods Applied to Time Series Forecasting"*. Conference Proceedings in Computational and Ambient Intelligence. 2007.

[12] Marco Cuturi and Arnaud Doucet. *"Autoregressive Kernels For Time Series"*. arXiv:1101.0673. 2011.

[13] Karl Mikalsen, Filippo Bianchi, Cristina Soguero-Ruiz and Robert Jenssen. *"Time series cluster kernel for learning similarities between multivariate time series with missing data"*. Pattern Recognition 76, 569-581. 2018.

[14] Huanhuan Chen, Fengzhen Tang, Peter Tino, and Xin Yao. *"Model-based kernel for efficient time series analysis"*. In Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining (KDD '13). 2013.

[15] Walter Rudin. *"Fourier analysis on groups."* Courier Dover Publications, 2017.

[16] Alexander M. Akhmetshin and I. V. Lyuboshenko. *"The reconstruction of signals and images from the noisy Fourier transform phase by means of the generalized difference principle."* In Pattern Recognition, 1996., Proceedings of the 13th International Conference on, vol. 2, pp. 370-375. IEEE, 1996.

[17] Hiroshi Shimodaira, Ken-ichi Noma, Mitsuru Nakai, and Shigeki Sagayama. "Dynamic time-alignment kernel in support vector machine." In Advances in neural information processing systems, pp. 921-928. 2002

[18] Marco Cuturi, Jean-Philippe Vert, Oystein Birkenes and Tomoko Matsui. *A Kernel for Time Series Based on Global Alignments*. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP '07). 2007.

[19] Alan V. Oppenheim. *"Discrete-time signal processing"*. Pearson Education India, 1999.

[20] Gabriel Wachman, Roni Khardon, Pavlos Protopapas, and Charles R. Alcock. *"Kernels for Periodic Time Series Arising in Astronomy"*. In Proceedings of the European Conference on Machine Learning and Knowledge Discovery in Databases: Part II (ECML PKDD '09), 2009.

[21] Charles Van Loan. *"Computational frameworks for the fast Fourier transform."* Vol. 10. Siam, 1992.

[22] Andrea Vedaldi and Andrew Zisserman. *"Efficient additive kernels via explicit feature maps."* IEEE transactions on pattern analysis and machine intelligence 34, no. 3 (2012): 480-492

[23] Yang, Jiyan, Vikas Sindhwani, Quanfu Fan, Haim Avron, and Michael W. Mahoney. *"Random laplace feature maps for semigroup kernels on histograms."* In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 971-978. 2014

[24] Avrim Blum. *"Random projection, margins, kernels, and feature-selection"*. In Proceedings of the 2005 international conference on Subspace, Latent Structure and Feature Selection (SLSFS'05). 2005.

[25] Ali Rahimi and Benjamin Recht. *"Random features for large-scale kernel machines"*. In Proceedings of the 20th International Conference on Neural Information Processing Systems (NIPS'07). 2007.

[26] Purushottam Kar and Harish Karnick. *"Random feature maps for dot product kernels."* In Artificial Intelligence and Statistics, pp. 583-591. 2012

[27] Ninh Pham and Rasmus Pagh. *"Fast and scalable polynomial kernels via explicit feature maps."* In Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining, pp. 239-247. ACM, 2013.

[28] Yanping Chen, Eamonn Keogh, Bing Hu, Nurjahan Begum, Anthony Bagnall, Abdullah Mueen and Gustavo Batista. *"The UCR Time Series Classification Archive"*. 2015. URL www.cs.ucr.edu/ eamonn/time_series_data/

[29] David Cox and Nicolas Pinto. *"Beyond simple features: A large-scale feature search approach to unconstrained face recognition."* Automatic Face Gesture Recognition and Workshops (FG 2011), 2011 IEEE International Conference on. IEEE, 2011.