Pamukkale Univ Muh Bilim Derg, 28(5), 681-691, 2022

Pamukkale Üniversitesi Mühendislik Bilimleri Dergisi

Pamukkale University Journal of Engineering Sciences

Investigation of Fourier features via neural networks and an application to smart steering in wireless mesh networks

Sınır ağları üzerinden Fourier özniteliklerinin incelenmesi ve kablosuz örgü ağlarda akıllı bağlantı yönlendirmeye uygulanması

Bulut KUŞKONMAZ[1*], *Hüseyin ÖZKAN*[1]

[1]Faculty of Engineering and Natural Sciences, Electronics Engineering, Sabancı University, İstanbul, Turkey.
kuskonmazbulut@sabanciuniv.edu, huseyin.ozkan@sabanciuniv.edu

Received/Geliş Tarihi: 06.02.2021 Revision/Düzeltme Tarihi: 10.01.2022 doi: 10.5505/pajes.2022.98371
Accepted/Kabul Tarihi: 18.01.2022 Research Article/Araştırma Makalesi

Abstract

Random Fourier features (RFF) provide one of the most prominent means for nonlinear classification in especially large scale data settings. However, considering the original proposal of RFF, Fourier features are randomly drawn from a certain distribution and used unoptimized. In this paper, we investigate Fourier features via a single hidden layer feedforward neural network (SLFN) and optimize, i.e., learn, those features (instead of drawing randomly). The learned Fourier features are deduced from the radial basis function (rbf kernel), and implemented in the hidden layer of the SLFN which is followed by the output layer. We present extensive experiments with 10 different classification datasets from various fields, e.g., bioinformatics. The learning of Fourier features is observed to be highly superior over the competing techniques such as perceptron in the rbf kernel space or a greedy forward feature selection strategy. On the other hand, the Fourier feature learning performs comparably with SVM (support vector machines with rbf kernel) while providing substantial computational benefits, and this is even without using the max margin regularization. Moreover, when tested in wireless mesh networks, the SLFN delivers promising smart steering capabilities.

Keywords: Fourier features, Neural networks, Single hidden layer, Classification, Kernel, Steering.

Öz

Rastgele Fourier öznitelikleri (RFÖ), doğrusal olmayan sınıflandırma için özellikle büyük ölçekli veri koşullarında en önemli araçlardan biridir. Bununla birlikte, RFÖ'nün orijinal önerisi dikkate alındığında, Fourier öznitelikleri belirli bir dağılımdan rastgele seçilir ve eniyilenmeden kullanılır. Bu yazıda, Fourier özniteliklerini tek gizli katmanlı bir ileri beslemeli sinir ağı (TKİS) aracılığıyla incelemekte ve bu öznitelikleri (rastgele seçim yerine) optimize etmekte, yani öğrenmekteyiz. Öğrenilen Fourier öznitelikleri radyal taban fonksiyonundan (rtf çekirdeği) üretildikten sonra TKİS'nin gizli katmanında gerçeklenir ve sonra takip eden çıktı katmanında kullanılır. Biyoinformatik gibi çeşitli alanlardan 10 farklı sınıflandırma veri kümeleri ile kapsamlı deneyler sunmaktayız. Fourier öznitelik öğrenmesinin, rtf çekirdek uzayında perseptron uygulama veya ileri yönlü fırsatçı öznitelikleri seçme stratejileri gibi rakip tekniklere göre oldukça üstün olduğu gözlemlenmiştir. Öte yandan, Fourier öznitelik öğrenmesi, DVM (rtf çekirdekli destek vektör makineleri) ile karşılaştırılabilir bir performans sergilerken, önemli hesaplama avantajlarını marjin büyütmesini kullanmadan dahi sağlayabilmektedir. Ayrıca, TKİS'yi kablosuz örgü ağlarında test ettiğimizde akıllı yönlendirme açısından umut vaat ettiğini gözlemlemekteyiz.

Anahtar kelimeler: Fourier öznitelikleri, Sinir ağları, Tek gizli katman, Sınıflandırma, Çekirdek, Bağlantı yönlendirme.

1 Introduction

Kernels in machine learning [1], e.g., support vector machines (SVM) [2],[3], enable the exploration of nonlinear data structures for various purposes such as classification [3], regression [4] and distribution modeling [5]. The central idea in learning with kernels is to map the observation space to a high dimensional kernel space by essentially changing the original inner product to a more appropriate one, i.e., kernel, providing only an indirect access to the high dimensional space. Afterwards, a linear technique is employed in the kernel space to solve a nonlinear problem in the observation space. There are two issues in this approach. First, an exploration of kernels is required to find the one that best fits to the nonlinearity in data; and in fact, there are a continuous spectrum of kernels that one can choose from. Second, the computational as well as space complexity are prohibitively complex when the data is abundant. The reason is that using the resulting kernel model requires (later after training) repetitive and complete (occasionally partial) passes over the data (that the model is trained on) since the kernel space is often not constructed explicitly and can be accessed only through the kernel. The first issue (finding the best kernel) can be addressed by kernel learning [6], and the second (prohibitive computational as well as space complexity) can be addressed by constructing the kernel space explicitly [7]. However, literature addressing the both simultaneously is fairly limited.

To this end, we consider a single hidden layer feedforward network (SLFN) for nonlinear binary classification. By using the random Fourier features (RFF) [7] of the rbf kernel, the hidden layer of the considered SLFN constructs (initially at the beginning) the high dimensional kernel space explicitly and compactly to achieve a direct access and low cost of mapping. The mapping to the kernel space is compact (the number of the hidden layer units is relatively small) thanks to the exponentially fast rate of approximation of the target kernel via RFF. Then, the output layer classifies in the constructed kernel space by linear means for nonlinear classification in the original observation space. We use stochastic gradient descent based

*Corresponding author/Yazışılan Yazar

training. As a result of the hidden layer parameter optimization during training, Fourier features (after they are randomly drawn at the beginning) are also learned, i.e., optimized, which yields the kernel learning aspect in the presented study. In this context, considering the nonconvex optimization framework of neural networks, we investigate various training strategies and compare with certain techniques, e.g., forward feature selection, via extensive experiments on 10 different datasets from several fields, e.g., bioinformatics. In our experiments, the presented approach of Fourier feature learning with neural networks is observed to significantly outperform the competitors. Additionally, we demonstrate an application to smart steering in wireless mesh networks. The presented SLFN is again superior in comparison to a previous study [8] which uses a perceptron in the kernel space for the same purpose.

In the following, we discuss the related work. Fourier features, first introduced in [7], are capable of approximately expanding (i.e. constructing) the high dimensional kernel space for any symmetric and shift invariant kernel when such features are appropriately randomly drawn from a certain probability distribution; and that distribution is given by the inverse Fourier transform of the kernel being used. The approximation improves exponentially fast with respect to the number of random Fourier features (RFF) while reaching up to an asymptotically exact -and importantly compact- expansion of the kernel space. Thus, at a relatively small cost for expanding the kernel space with RFF, it is possible to significantly speed up both the training and test phases of kernel machines with computation as well as space wise huge advantages. This has led in literature to an extensive exploration into large scale data processing based on RFF. For example, an online algorithm has been proposed in [9] for nonlinear classification in large scale settings, which is similar to a perceptron in the kernel space constructed by RFF. That online perceptron in kernel space has also been used successfully, in [8], for managing the connections between the clients and access points in a wireless mesh networks. In these studies, Fourier features are randomly drawn as independent of the data and used unoptimized.

Observing the correspondence between the Fourier feature sampling distribution and the kernel being approximated, optimizing the distribution directly in Fourier domain has been studied in [10] and in [11] to learn parametric kernels in a data driven manner from a certain family, e.g, rbf kernels. The difficulty due to the need of resampling at each time the sampling distribution is manipulated has been overcome in both studies by the re-parametrization of a predetermined set of features from a mother distribution in the same family. Therefore, the outcome is the learning of the hyperparameters of a kernel, rather than learning of a nonparametric custom kernel. For instance, a Gaussian kernel can be learned unisotropically with different variances at each feature dimension. Note that the study [10] can be seen as a generalization of the study [11] where the latter -though- provides relatively deeper theoretical insights. On the other hand, the approach in [12] is different in the sense that the sampling distribution is modeled as a mixture of Gaussians in Fourier domain with a Dirichlet process prior and then learned in a data driven Bayesian manner. Consequently, as an advantage, the class of kernels of the study [12] that the technique learns in is larger compared to the one of [10],[11]. Instead of explicitly modeling the sampling distribution whether in a parametric [10],[11] or nonparametric manner [12], one can approach the kernel learning problem in a rather generative manner as in [13]. In this generative approach, samples from a known base distribution are transformed by, for instance, a neural network and then the transformed samples are treated as the samples from the unknown sampling distribution in the Fourier domain. Learning of that neural network allows to access to kernel evaluations without needing to know the sampling distribution explicitly, which leads to a generative kernel learning approach [13] that is studied in the context of generative deep learning and supervised models.

A completely different approach is presented in [14]. In that approach, instead of learning the distribution (whether in a parametric or nonparametric or generative manner) in Fourier domain, Fourier features are directly learned in a data driven manner with respect to a certain loss such as regression or classification. The sample of the learned Fourier features at the end of the training can perhaps be regarded as the learned sampling distribution in the Fourier domain, which does not necessarily correspond to the inverse Fourier transform of a symmetric and shift invariant kernel. On the other hand, such learned Fourier features define a new inner product which constitutes the kernel learning aspect in [14], where the kernel implied by the learned Fourier features is a custom one that is not restricted to the class of symmetric and shift invariant kernels. In this line of research, a two-layer or three-layer neural network has been trained with stochastic gradient descent optimization in [14], and compared with the original RFF method of [7]. This learning of Fourier features with neural networks is considered in [15],[16] as an RFF layer and multiple such layers are cascaded end-to-end to achieve deeper architectures, where each layer can essentially learn a different kernel. The same RFF layer as an SLFN is studied in [17] for the completely different goal of Neyman-Pearson classification to achieve false alarm rate controllability. Instead of using an RFF layer in a neural network, one can also learn Fourier features by using a forward feature selection method. For instance, in [18], a pool of Fourier features is maintained and expanded iteratively. A single feature is newly designed and added to the pool at each iteration such that the hard examples of the previous iteration are better classified. Iterations end when the desired number of iterations (or a sufficiently small regression loss) is reached. This approach is an alternative to the neural network based Fourier feature learning.

In contrast to the studies [7]-[9] of using Fourier features in random as independent with data; in our presented study, Fourier features are learned in a data driven manner and hence the disadvantage of using RFF unoptimized and random is removed. Learning of Fourier features does also differentiate our study from the studies [10],[11] and [12],[13], where the sampling distribution is learned in Fourier domain that define Fourier features whereas we learn Fourier features that define the sampling distribution. On the other hand, the neural network based Fourier feature learning technique in our presented study is conceptually the same as the SLFN of the report [14],[17] or the RFF layer of the conference proceedings [15],[16]. However, one difference is that the phase of the sinusoid of the Fourier feature is obtained as a linear combination of sine and cosine terms in [14],[15],[17]; hence, a bias term in their neural network is skipped. Whereas the phase is directly defined as the bias term in our neural network and in the one of [16]. This has two outcomes in favor of our presented study. First, A) the size of the hidden layer in [14],[15],[17] is twice as larger as the one of our network; namely, our network is more compact providing computation as well as space wise

advantages. Second, B) our network learns one unbounded weight magnitude and one bounded (in the interval $[0, 2\pi]$]) bias term compared to the two unbounded weight magnitudes to be learned in [14],[15],[17]. This provides another advantage to our network structure in terms of the training and weight estimation. The same advantages hold for the study [16] as well. Note that [17] considers a completely different goal of false positive rate controllability.

In this respect, the contributions of our study can be summarized as follows. First, we emphasize that the goal of the presented study is not to propose a new Fourier feature learning technique. Instead, as an important contribution, we comprehensively analyze an existing technique of Fourier feature learning (cf. [14]-[17]) from the perspective of several training strategies and investigate Fourier features via neural networks in deeper detail. In this sense, our presented study comprehensively extends the works in the previous reports and conference proceedings [14]-[17]. Additionally, we demonstrate the efficacy of Fourier features learning with neural networks in a real life application to smart steering in wireless mesh networks. In particular, our aim is to concentrate on various neural network training strategies such as the stochastic gradient based optimization (this is used to obtain computation and space-wise efficient algorithms) as well as a coordinate descent type optimization. In the latter one, iterations are conducted in two ways: epoch-by-epoch (each epoch is a pass over the data) and minibatch-by-minibatch (each minibatch can be as small as a single instance). We also consider a variant of the feature selection of [18] as a baseline to analyze the compactification power of the Fourier feature learning with SLFN. Furthermore, the original RFF technique presented in [7] is also considered along with SVM as two other baselines. We compare these training strategies and techniques based on a comprehensive experimental analysis with 10 different datasets from various fields such as genetics, and the best performing ones are identified. Note that our experiments are significantly more comprehensive in terms of the training methodology, compared to the conference proceedings or technical reports [14]-[17]. Moreover, we demonstrate the presented approach (i.e. Fourier feature learning with neural networks) based on a real life challenging application to smart client steering for connection management in wireless mesh networks. In this application, the presented approach is significantly superior over the recently published results in [8].

The goal of the presented work and the background about random Fourier features are provided in Section 2 and Section 3, respectively. In Section 4, Fourier feature learning is presented, and then Section 5 presents our experimental evaluation along with a demonstration of the presented approach for smart steering in wireless mesh networks. We conclude in Section 6 with final remarks.

2 Goal of the presented work

We study nonlinear binary classification (e.g. recognizing a visual object as a car or human) for any given set of data represented by $\{(x_t, y_t)\}_{t=1}^{N}$. In the presented study, a data instance $x_t \in R^{d\times 1}$ is a dimensional observation vector, $y_t \in \{-1,1\}$ is a binary valued corresponding label and N is the number of observations. In order to achieve nonlinear modeling capability, we use the kernel approach to nonlinear classification. In this approach, a kernel function $k(\cdot,\cdot)$ encodes the inner product between any two instances x_i and x_j in a high dimensional space, where $R^{D\times 1} \ni z = \phi(x)$ is the mapping to the high dimensional space with $z_i^T z_j = k(x_i, x_j)$ [19],[20]. This mapping of the kernel approach can be considered as the transformation of the nonlinear data manifold in the observation x space with the kernel similarity into a Euclidean high dimensional z space with the inner product similarity. Consequently, one can simply apply a linear classifier in high dimension to solve a nonlinear classification problem in the original space. This is also known as the kernelization of linear techniques or simply ``kernel trick", cf. Figure 1 for a visual interpretation. Furthermore, an appropriate kernel function in hand is typically sufficient to exploit the power of kernels without constructing the mapping $\phi(\cdot)$ explicitly. This perhaps provides a conceptual advantage, which -however- leads to a computational drawback. For example, if we consider the classification function (with γ and β being the classifier parameters) $\delta(x) = \sum_{i=1}^{N_{sv}} \gamma_i\, k(x, x_i) + \beta$ of the kernelized support vector machines (SVM), it is straightforward to observe that the computational complexity (in the test phase) is $O(N_{sv})$ and the number of support vectors N_{sv} can be as large as the size of the training set. This is prohibitively complex, and thus hinders real time processing in especially the contemporary fast streaming applications that constantly present data in large scales. Similar issues appear in the training phase as well, since training is typically more complex than testing and then the cost of using kernels folds more harshly in large scale data conditions. Therefore, constructing the mapping $\phi(\cdot)$ explicitly appears to be the key to designing techniques that are computationally efficient while benefiting the power of kernels. To this end, we consider the kernel approach to binary classification and particularly concentrate on random Fourier features [7],[9] for an explicit construction of the kernel space. However, random Fourier features are -in its original proposal [7]- independent of data. For this reason, our goal is to investigate various training approaches and algorithms for the learning of Fourier features in the context of neural networks. For this purpose, we present a comprehensive set of experiments with 10 different benchmark datasets. In addition, we demonstrate an application of the learned Fourier features to smart steering (by using the data of [8]) in wireless mesh networks, where we achieve significantly superior performance compared to the method in [8].

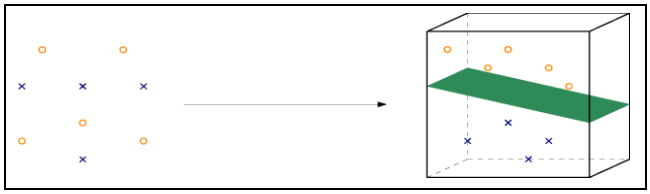

Figure 1. Visual interpretation of the kernel trick for nonlinear classification with linear techniques. The original observation space (left) is transformed into the high dimensional kernel space (right) via the mapping $\phi(\cdot)$.

3 Random fourier features

Random Fourier features (RFF) [7] provide a means to compactly approximate a symmetric and shift invariant kernel function, which can be used to achieve computationally substantial gains in applications of classification with kernels. A properly scaled symmetric and shift invariant kernel function $k(x_i, x_j)$ can be written as a single argument function: $k(x_i, x_j) = k(x_j, x_i) = f(x_i - x_j) = f(x_j - x_i)$, as explained in the study [7] by Rahimi and Recht. Then, by Bochner's theorem

[7],[21], inverse Fourier transform of $f(\cdot)$ yields a proper probability density function $p(w)$, i.e., $k(x_i, x_j) =$

$$f(x_i - x_j) = \int_{R^d} p(w)\, cos\left(w^T(x_i - x_j)\right) dw$$

$$= E_w\left[[cos(w^T x_i), sin(w^T x_i)]\left[cos\left(w^T x_j\right), sin\left(w^T x_j\right)\right]^T\right]$$

$$\simeq \overline{z_i^T} z_j,$$

where $p(\cdot)$ is real since $f(\cdot)$ is real and even, $E_w(\cdot)$ is the expectation with respect to $p(w)$,

$$R^{2D} \ni \bar{z}_i = \left(1/\sqrt{D}\right)[cos(w_1^T x_i), sin(w_1^T x_i), \cdots, cos(w_D^T x_i), sin(w_D^T x_i)]^T \quad (1)$$

is the random mapping to the $2D$ dimensional kernel space with $\{w_r\}_{r=1}^D$ being an i.i.d. sample drawn from $p(w)$ (similarly for $\bar{z}_j$ of x_j and for $\bar{z}$ of x in general), and the last approximate equation is due to the law of large numbers. Here, $\bar{z}$ explicitly and randomly constructs the high dimensional kernel space in which the inner product approximates the kernel function in the original space. Therefore, as a first remark, this is a compact mapping since the approximation improves exponentially fast with respect to the dimension D (of the high dimensional space) due to the Hoeffding's inequality [7]; hence, a relatively small D is expected to perform well. Second, one can solve a nonlinear problem in the original observation space by a linear technique in the high dimensional space at the cost of the mapping. Finally, third, one can also use the mapping

$$R^D \ni z_i = \left(\sqrt{2/D}\right)[cos(w_1^T x_i + b_1), \cdots, cos(w_D^T x_i + b_D)]^T \quad (2)$$

(with $\{b_r\}_{r=1}^D$ being an i.i.d. sample drawn from uniform distribution $U(0,2\pi)$ in $[0,2\pi]$) for the same purpose, i.e., $k(x_i, x_j) \simeq z_i^T z_j$, since there exists $\alpha \in R^D$ corresponding to any $\bar{\alpha} \in R^{2D}$ such that $\overline{z^T}\bar{\alpha} = z^T$ which can be straightforwardly observed by phasor addition [22].

Considering our previous example, the decision function of the kernelized SVM $\delta(x) = \sum_{i=1}^{N_{sv}} \gamma_i\, k(x, x_i) +$ can now be approximated as $\delta(x) \simeq \sum_{i=1}^{N_{sv}} \gamma_i\, z^T z_i + \beta = z^T\left(\sum_{i=1}^{N_{sv}} \gamma_i\, z_i\right) + \beta = z^T\alpha + \beta$, where $\alpha = \sum_{i=1}^{N_{sv}} \gamma_i\, z_i$. This random mapping with RFF provides substantial gains as the computational complexity shrinks down to $O(1)$ (from the complexity $O(N_{sv})$ of the kernelized SVM) for testing an instance at the computational cost $O(D)$ of the random mapping $x \to z = \phi(x)$. Furthermore, this random mapping with RFF does also allow online processing (one example is presented in [9]) for large scale data in real time while maintaining the nonlinear modeling capability, with again the complexity $O(1)$ per instance.

We next continue with the learning of Fourier features in order to remove the randomization and design a data driven method.

4 Learning fourier features

Random Fourier features (RFF) $(w_r, b_r)_{r=1}^D$ as an i.i.d. sample drawn from $p(w) \times U(0,2\pi)$ are indeed powerful features enabling computationally highly efficient nonlinear classification for large scale data in real time. However, an improvement is certainly possible by learning such features in a data driven manner, as opposed to relying on a random sample drawn without taking into account the data. To this end, we observe that the RFF based random mapping in (2) can be implemented in the first and hidden layer of a network with the parameters w and b, and then the output layer follows with the parameters α and β. The result is a single hidden layer feedforward neural network (SLFN) as visually represented in Figure 2, where the first and hidden layer includes D units and the corresponding parameters are randomly initialized as $(w_r, b_r)_{r=1}^D \sim N(0,2gI) \times U(0,2\pi)$, and thus the set of random Fourier features (RFF), i.e., $\{cos(w_r^T x_t + b_r)\}_{r=1}^D$, is the set of hidden layer activations with the sinusoidal activation function $cos(\cdot)$. In this work, we use the radial basis function (rbf) $k(x_i, x_j) = exp(-g\left||x_i - x_j\right||^2)$ as the kernel, where g is the bandwidth parameter and hence the randomization is Gaussian given by the inverse Fourier transform the rbf kernel: $p(w) = N(0,2gI)$ and I is the $d \times d$ identity matrix. The presented work can be straightforwardly extended to any symmetric and shift invariant kernel.

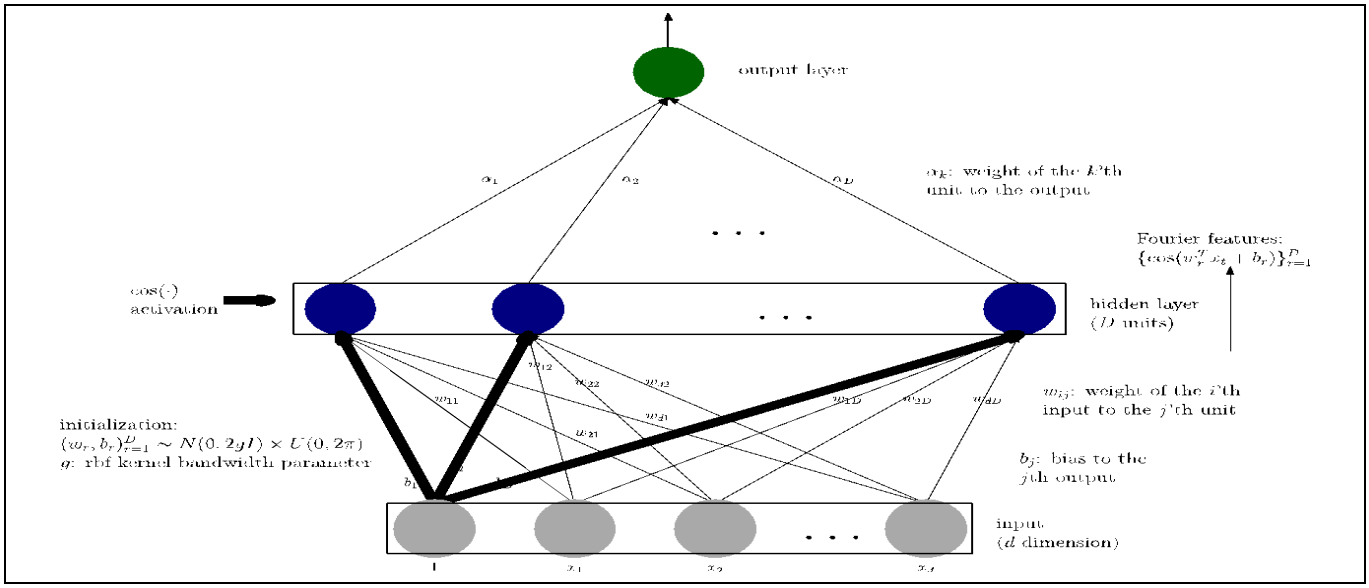

Figure 2. Visual representation of our single hidden layer feedforward neural network (SLFN), where D is the size of the hidden layer and d is dimension of the data. The hidden layer activation is chosen sinusoidal to produce the Fourier features $\{cos(w_r^T x_t + b_r)\}_{r=1}^D$; and initially, $(w_r, b_r)_{r=1}^D$ are randomly drawn from the density $N(0,2gI) \times U(0,2\pi)$.

We emphasize that even if the hidden layer of this SLFN Figure 2 is kept untrained, the network is still expected to perform well. This is because the hidden layer is designed and initialized to approximately expand the kernel space, in which the linear classification can already model almost any nonlinearity in the original space (provided that the kernel being exploited is appropriate). Thus, RFF also provides a decent initalization to the subsequent training phase. On the other hand, training the hidden layer optimizes the weights $\{(w_r, b_r)\}_{r=1}^{D}$, which yields the learning of Fourier features at the hidden layer activations $\{cos(w_r^T x_t + b_r)\}_{r=1}^{D}$.

4.1 Various training approaches for learning Fourier features

This section provides our training approaches which can be categorized as classification with A) random Fourier features (single layer learning: online kernel perceptron or a perceptron trained in the rbf kernel space, as well as SVM with rbf kernel), B) SLFFS: single layer learning with selected Fourier features (forward selection), C) SLFN in the typical training settings, and D) SLFN with coordinate descent type optimization. In those approaches, we use backpropagation together (when it applies) with SGD or minibatch optimization as well as cross entropy (CE) or mean square error (MSE) losses.

4.2 Single layer learning (SL)

This approach -as a baseline for our comparisons- keeps the hidden layer untrained and only learns the output layer, which essentially implements a kernel machine to obtain a classifier in the kernel space. One can use here stochastic gradient descent (SGD) or minibatch for training: both correspond to principally the same large scale online kernel perceptron in [9] (i.e. a perceptron trained in the rbf kernel space) or the smart steering classifier in [8]. In addition, SVM with the rbf kernel (or linear SVM in the kernel space expanded by random Fourier features [7],[23]) also falls in this category, since a margin based classifier is trained in the kernel space without an attempt to optimize the kernel space.

4.2.1 Single layer learning with fourier feature selection (SLFFS)

This approach -as another baseline for our comparisons- follows an alternative to the neural network based Fourier feature learning, and it can be seen similar in nature to the feature combination with boosting presented in [18]. In this approach, we randomly draw a large set, i.e., pool, of Fourier features as previously described, and then select the useful Fourier features (from that initial large set, i.e., pool) in a greedy manner with forward selection based on a certain criterion, e.g., mean square error. This approach is to compare the compactification power of Fourier feature learning, i.e., to investigate whether the SLFN or feature selection performs favorably with the same number of Fourier features.

4.2.1.1 Two Layer Learning (TL)

This approach is the typical neural network training setting with SGD or minibatch optimization [19]. Both the hidden and output layers are trained.

4.2.2 Batch-Based two layer learning (TL-B)

This training approach processes the data minibatch by minibatch iteratively in a coordinate descent type optimization framework [24]. One iteration learns the output layer while keeping the hidden layer untrained, and the following iteration learns the hidden layer while keeping the output layer untrained. Iterations follow each other in an alternating manner. Each minibatch can be chosen as small as a single data instance or as a tiny subset.

4.2.3 Epoch-Based two layer learning (TL-E)

This training approach is actually the batch-based two layer training, where each minibatch is chosen as the complete training dataset. In this case, we call iterations as epochs and each epoch is a complete pass over the data. This is to better investigate the coordinate descent type optimization framework [24] with more robust derivatives.

5 Experiments

Our experiments extensively investigates Fourier features based on the aforementioned training approaches to binary classification. In particular, we first conduct a performance analysis with 10 different benchmark datasets from various fields, and then demonstrate an application to steering in wireless mesh networks where we achieve superiors results compared to a recent study [8].

5.1 Investigation with Benchmark Datasets

The benchmark of 10 classification datasets that we use in this part can be found in [25] and summarized in Table 1. Each dataset is z-score standardized (zero mean and unit variance) before the processing, and shuffled afterwards to obtain 10 different permutations. Also, 5-fold cross validation is used for parameter optimization and in each case, 80% (20%) is reserved for training (testing). Mean accuracy results are reported across 10 different permutations along with the corresponding standard deviations. Optimized parameters are the dimension of the kernel space $D \in \{0.5, 1, 2, 4, 8, 10, 20, 30, 50, 100\} \times d$ (where d is the data dimension and we use ceiling when necessary to round to integer) as well as the kernel bandwidth parameter $g \in \{0.01, 0.02, 0.04, 0.08, 0.15, 0.3, 0.5, 1, 2, 4, 8, 10, 20, 30, 40, 50\}$. Minibatch size is 100 and the learning rate is 0.01 in all of the experiments.

Based on our overall classification results that are reported in Table 2, our observations are as follows: 1) cross entropy (CE) loss yields better results compared to the loss of mean square error, 2) TL algorithm generally outperforms the others, and 3) using minibatch or SGD for optimization seem to not generate a significance difference. Consequently, SLFN based learning of Fourier features is superior over a plain kernelization (cf. the comparisons between TL's and SL) and observed to be promising in terms of enabling computationally efficient online processing due to the comparability (in terms of accuracy) SGD and minibatch. Also, a joint learning of Fourier features and classifier is observed to outperform the coordinate descent type learning (cf. the comparisons between TL and TL-B or TL-E). Therefore, we continue our experiments below with the TL algorithm trained based on the CE loss since it is observed to outperform the others. Our benchmark results in Table 2 are produced by using a similar setting for all algorithms for fairness. After choosing the best performing TL algorithm (with CE and SGD or minibatch) as discussed above, we now further optimize it (TL algorithm with CE and SGD or minibatch) standalone in terms of the number of minibatches and learning rate while comparing to SVM with rbf kernel.

Table 1. Benchmark details as provided in [25].

Dataset	Dimension	# of Instances (+,-)	# of epochs, minibatches/SGD	Data type
Australian	14	650 (363, 287)	300/20	Real/Australian Credit Approval
Banana	2	5000 (2769, 2231)	250/10	Synthetic
Breast Cancer	10	600 (375, 225)	300/20	Real/Diagnostic Wisconsin Breast Cancer Database
Diabetes	8	750 (263, 487)	250/20	Real/Pima İndians Diabetes Dataset
Fourclass	2	850 (547, 303)	300/20	Synthetic
German Numer	24	1000 (700, 300)	150/10	Real/German Credit Data
Phishing	68	11000 (4877, 6123)	10/2	Real/Phishing Website Dataset
Splice	60	3000 (1454, 1546)	60/10	Real/Splice Junctions in DNA Sequence
Svmguide1	4	7000 (3054, 3946)	150/10	Synthetic
Svmguide3	21	1250 (919, 331)	150/10	Synthetic

Table 2. Benchmark results (mean accuracy over 10 different data permutations along with the corresponding standard deviation) of TL, SL, TL-E and TL-B algorithms with CE/MSE loss and minibatch/SGD optimizer on ten different datasets.

	TL (CE)	SL (CE)	TL-E (CE)	TL-B (CE)	TL (MSE)	SL (MSE)	TL-E (MSE)	TL-B (MSE)
Australian	SGD **87.23 ± 2.90** minibatch 87.07 ± 1.27	SGD 84.07 ± 3.60 minibatch 85.15 ± 2.17	SGD 84.92 ± 4.65 minibatch 86.84 ± 2.74	SGD 86.23 ± 2.92 minibatch. 86.07 ± 2.74	SGD 72.38 ± 16.93 minibatch 69.84 ± 15.37	SGD 70.00 ± 16.92 minibatch 60.15 ± 11.11	SGD 67.84 ± 15.57 minibatch 69.00 ± 14.15	SGD 66.92 ± 14.75 minibatch 63.76 ± 13.03
Banana	SGD 89.32 ± 1.14 minibatch **89.82 ± 0.66**	SGD 88.45 ± 1.53 minibatch 89.48 ± 0.65	SGD 89.22 ± 1.11 minibatch 89.35 ± 1.21	SGD 89.68 ± 1.19 minibatch 89.19 ± 1.34	SGD 89.15 ± 1.11 minibatch 85.92 ± 5.59	SGD 87.99 ± 1.28 minibatch 85.59 ± 3.68	SGD 87.83 ± 1.11 minibatch 83.41 ± 3.57	SGD 88.53 ± 1.27 minibatch 84.86 ± 3.07
Breast Cancer	SGD 95.58 ± 1.75 minibatch 95.83 ± 1.90	SGD 94.58 ±1.63 minibatch 95.16 ± 1.61	SGD 95.66 ±1.74 minibatch **96.25 ± 1.25**	SGD 95.66 ± 1.74 minibatch 95.75 ± 1.80	SGD 68.66 ± 25.47 minibatch 73.16 ± 21.96	SGD 75.25 ± 28.01 minibatch 51.50 ± 20.78	SGD 71.91 ± 25.04 minibatch 59.91 ± 22.18	SGD 76.83 ± 24.44 minibatch 51.41 ± 20.16
Diabetes	SGD **76.46 ± 2.61** minibatch 76.40 ± 1.40	SGD 73.73 ± 3.54 minibatch 73.40 ± 3.14	SGD 74.66 ± 4.39 minibatch 74.86 ± 2.89	SGD 75.00 ± 3.05 minibatch 74.40 ± 3.22	SGD 72.13 ± 6.10 minibatch 63.55 ± 12.58	SGD 61.53 ± 15.66 minibatch 73.55 ± 4.01	SGD 69.06 ± 11.95 minibatch 54.55 ± 13.25	SGD 61.13 ± 18.16 minibatch 63.33 ± 7.68
Fourclass	SGD **99.88 ± 0.23** minibatch 98.41 ± 1.14	SGD 99.47 ± 0.61 minibatch 96.23 ± 1.98	SGD 99.70 ± 0.54 minibatch 96.88 ± 1.53	SGD 99.70 ± 0.47 minibatch 96.88 ± 1.53	SGD 96.00 ± 4.71 minibatch 69.58 ± 17.26	SGD 92.23 ± 8.24 minibatch 64.00 ± 14.80	SGD 93.82 ± 5.87 minibatch 69.52 ± 14.54	SGD 95.35 ± 3.60 minibatch 69.58 ± 14.56
German Numer	SGD 73.90 ± 3.76 minibatch **74.75 ± 2.96**	SGD 72.85 ± 2.96 minibatch 68.45 ± 3.55	SGD 72.90 ± 4.18 minibatch 67.90 ± 4.53	SGD 73.50 ± 3.15 minibatch 70.15 ± 3.83	SGD 67.50 ± 13.76 minibatch 57.20 ± 19.13	SGD 61.30 ± 13.82 minibatch 59.00 ± 16.07	SGD 58.90 ± 19.83 minibatch 50.30 ± 18.75	SGD 66.05 ± 13.79 minibatch 48.80 ± 19.73
Phishing	SGD **93.52 ± 0.73** minibatch 92.75 ± 0.61	SGD 92.31 ± 1.08 minibatch 89.90 ± 1.19	SGD 93.34 ± 0.63 minibatch 92.89 ± 0.87	SGD 90.85 ± 5.23 minibatch 92.37 ± 1.81	SGD 77.85 ± 18.16 minibatch 57.28 ± 15.48	SGD 83.06 ± 13.63 minibatch 59.66 ± 8.68	SGD 84.60 ± 13.82 minibatch 68.43 ± 10.20	SGD 83.76 ± 14.11 minibatch 71.40 ± 10.28
Splice	SGD **84.05 ± 1.10** minibatch 83.88 ± 1.24	SGD 78.60 ± 2.64 minibatch 50.05 ± 5.72	SGD 79.10 ± 7.57 minibatch 80.35 ± 2.58	SGD 80.50 ± 6.67 minibatch 80.11 ± 1.98	SGD 72.85 ± 12.90 minibatch 62.06 ± 10.97	SGD 71.83 ± 6.93 minibatch 53.91 ± 4.97	SGD 73.40 ± 9.75 minibatch 59.03 ± 6.58	SGD 72.28 ± 10.73 minibatch 58.80 ± 6.71
Svmguide1	SGD 96.35 ± 0.45 minibatch **96.55 ± 0.37**	SGD 95.60 ± 0.61 minibatch 95.78 ± 0.83	SGD 95.76 ± 0.65 minibatch 95.21 ± 1.48	SGD 96.38 ± 0.43 minibatch 95.75 ± 0.56	SGD 95.70 ± 0.88 minibatch 82.80 ± 18.30	SGD 90.49 ± 12.43 minibatch 85.01 ± 17.88	SGD 95.10 ± 1.20 minibatch 84.95 ± 12.50	SGD 90.20 ± 12.46 minibatch 82.32 ± 15.15
Svmguide3	SGD **80.72 ± 3.03** minibatch 79.28 ± 3.10	SGD 77.60 ± 3.95 minibatch 72.92 ± 2.70	SGD 75.52 ± 10.58 minibatch 78.44 ± 2.48	SGD 77.08 ± 9.69 minibatch 76.72 ± 3.37	SGD 69.24 ± 18.13 minibatch 62.64 ± 18.09	SGD 66.00 ± 19.58 minibatch 67.32 ± 15.08	SGD 70.08 ± 7.92 minibatch 68.04 ± 7.20	SGD 73.20 ± 12.86 minibatch 65.80 ± 12.60

The resulting accuracy performance is given in Table 3, where the kernel bandwidth parameter g and the number D of Fourier features are used as the previously cross validated choices. We observe that the TL algorithm now performs comparable with (or slightly outperforms in 8 cases out of 10) SVM with rbf kernel. This is different from our previous observation in which the TL algorithm significantly outperforms (in contrast to the comparability or slightly better performance in favor of the TL algorithm in Table 3 compared to SVM with rbf kernel) the SL algorithm (cf. Table 2). On the contrary, we point out that SVM with rbf kernel and SL algorithm are in fact similar in principle (as they both do not attempt to optimize the given kernel space) while having two important differences: A) SVM with rbf kernel additionally incorporates the strong max margin concept as a regularizer [23] (our TL or SL algorithms do not have the max margin regularizer) and B) SVM with rbf kernel assumes access to all data all the time whereas our TL and SL algorithms are both sequential in nature with access to only a single instance or a minibatch each time. Therefore, we consider that the use of max margin regularization in SVM with rbf kernel along with complete data access (to all data all the time) explains this difference between our observations and also explains the exception in the case of Splice dataset (the only relatively low performance of the TL algorithm compared to SVM with rbf kernel, cf. Table 3).

Table 3. Comparison of TL algorithm (CE loss) with SVM (rbf kernel) in terms of mean accuracy (over 10 different data permutations) along with the corresponding standard deviations.

Data (epochs, earning rate)	TL mean $\pm$ std	SVM mean $\pm$ std
Australian (500, 0.008)	**87.76 $\pm$ 1.70**	86.00 $\pm$ 2.13
Banana 400, 0.01)	**90.08 $\pm$ 0.45**	90.04 $\pm$ 0.75
Breast Cancer (500, 0.01)	**96.33 $\pm$ 1.58**	96.16 $\pm$ 1.76
Diabetes (550, 0.005)	**76.93 $\pm$ 2.17**	76.26 $\pm$ 1.94
Fourclass (900, 0.01)	**99.94 $\pm$ 0.17**	99.88 $\pm$ 0.24
German Numer (350, 0.01)	**76.60 $\pm$ 2.47**	76.10 $\pm$ 3.00
Phishing (200, 0.01)	94.37 $\pm$ 0.47	**96.95 $\pm$ 0.36**
Splice (900, 0.001)	84.83 $\pm$ 1.34	**91.03 $\pm$ 1.20**
Svmguide1 (350, 0.05)	**96.84 $\pm$ 0.34**	96.79 $\pm$ 0.35
Svmguide3 (250, 0.01)	**81.08 $\pm$ 2.99**	81.06 $\pm$ 3.07

Importantly, due to the comparable performance between the TL algorithm and SVM with rbf kernel, we observe that learning Fourier features (TL algorithm) is highly effective as it is able to compete with the strong max margin regularization, and this is even in the sequential setting with online processing (recall that SVM with rbf kernel assumes access to all data all the time whereas our TL algorithm is sequential in nature with access to only a single instance or a minibatch each time). Therefore, one can certainly expect to -even sequentially- significantly outperform SVM with rbf kernel by using the max margin regularization based on learning Fourier features (for instance, in conjunction with the TL algorithm). As a result, we conclude that learning of Fourier features (our TL algorithm) is largely beneficial based on our presented comparisons and analyses. This substantial benefit is not only performance-wise, also computationally. We stress that the presented learning of Fourier features in the context of SLFN and the resulting sequential classification (TL algorithm) is computationally highly efficient (as described in detail in the previous Section 2 and Section 3), cutting down the computational complexity of SVM with rbf kernel from $O(N^3)$ to $O(N)$ in training and from $O(N^2)$ to $O(N)$ in test, where N is the number of processed instances (The number of support vectors in SVM can be as large as the number of training instances).

Our last experiment in this section is to compare, for Fourier feature learning, A) the introduced SLFN (TL algorithm) and B) the SL algorithm with forward feature selection (SLFFS). In our context, the SLFFS algorithm provides an alternative Fourier feature learning method as it greedily (and in a forward manner) chooses the most informative Fourier features from an initial large set. Therefore, we use the comparison between the algorithms TL and SLFFS to demonstrate the efficacy of Fourier feature learning via neural networks, cf. the SLFN design in Figure 2 that is used in the TL algorithm.

Recall that the TL algorithm learns and exploits D Fourier features. Therefore, for a fair comparison, we design the SLFFS algorithm to select the same number D of Fourier features from an initial large random pool consisting of $10D$ randomly drawn Fourier features. The selection process is greedy and operates in a forward manner with respect to a certain criterion. For instance, this criterion in [18] is the weighted least squares (LS) regression in the framework of boosting, where the weights are over data samples and a Fourier feature is designed in a specific manner at each iteration (to well separate the sampled pair of positive and negative instance of that iteration). We also use the LS regression to select Fourier features; however, our selection is out of a random large pool based on the LS regression loss whereas the selection in [18] is essentially a specific combination by boosting (cf. [18] for the details). The forward greedy selection in our SLFFS is as follows. In the first step, we obtain a Fourier feature from the pool, fit to the classification labels via LS regression, and record the mean square error (MSE). This is repeated for every single feature in the pool, then we select at the end our first Fourier feature (which minimizes the MSE) as a result of the first step and eliminate the selected feature from the pool. In any step, we obtain a feature from the pool, temporarily concatenate it to the features that are previously chosen, use the LS fitting and record the MSE. This is repeated for every single feature in the pool and then we select at the end our next Fourier feature (which minimizes the MSE) as a result of that step and eliminate the selected feature from the pool. This process continues until we select D Fourier features. We have three remarks regarding the described selection. A) Instead of selecting one by one, we select the Fourier features ten by ten to speed up the process, B) the kernel bandwidth parameter g and the number D of Fourier features are used as the previously cross validated choices, and C) note that we have ten random permutations for each dataset, each permutation has training and test sets, and the described feature selection here is completed within the training set based on 5-fold cross-validation. After this selection is completed, we initialize the first layer of our SLFN in Figure 2 with the selected features and then run the SL algorithm. Hence, in this experiment, the SL algorithm with the selected features (i.e. Fourier feature learning with a greedy forward selection) competes with the TL algorithm (i.e. Fourier feature learning with a neural network) as a comparison. The SL algorithm with

the selected features is called in the presented work as the FFS algorithm.

We report our findings in Table 4. In this table, the results of the SLFFS algorithm are newly presented, whereas the results of the algorithms TL and SL are copied from Table 2 to have them all in one place for ease of exposition. Here, we do not opt to include the improved results of the TL algorithm from Table 3 (instead we use Table 2) in order to have a fair comparison between TL and SL. Based on our findings, it is strongly observed that the algorithm SLFFS outperforms the algorithm SL, and the TL algorithm significantly (and also uniformly across all datasets) outperforms the algorithm SLFFS. The observation that both of TL and SLFFS yield a better performance in comparison to SL reinforces once again the effectiveness of the learning of Fourier features compared to the random use as originally proposed in [7]. On the other hand, since TL is significantly superior over SLFFS, we conclude that the learning of Fourier features with the introduced SLFN stands out as a reliable learning technique compared to the alternative of forward selection. One interesting observation is in the case of Splice dataset, where the algorithms SL and SLFFS appear to fail whereas the TL algorithm (learning of Fourier features with the introduced SLFN) is again highly successful.

This can be attributed to the large dimensionality of this dataset despite the low number of instances as well as the intrinsic linear structure of this dataset. In addition to large dimensionality, since the random Fourier features in SL and the selection of Fourier features from a random set in SLFFS are both incapable of generating a linear model out of a kernel approach, SL and SLFFS appear to be not performing satisfactorily in this case of Splice dataset. Another interpretation is that starting with a nonlinear model to solve a linear problem is typically inefficient, especially under harsh conditions due to the large ratio of the dimension to number of instances. On the other hand, the gradient descent based learning of Fourier features in the TL algorithm is able to successfully steer from an originally nonlinear model to a linear model, and thus TL significantly outperforms. We emphasize that the TL algorithm readily and successfully solves the nonlinear classification problems as well, which can be clearly observed (by the decent performance figures of TL uniformly across all datasets) from the classification results in Table 2, Table 3 and Table 4.

Table 4. Comparison of the algorithms TL, SL and SLFFS (SL algorithm applied with the selected Fourier features). Mean accuracy across 10 different data permutations is reported.

Data	TL mean $\pm$ std	SL mean $\pm$ std	SLFFS mean $\pm$ std
Australian	**87.07 $\pm$ 1.27**	85.15 $\pm$ 2.17	86.30 $\pm$ 3.01
Banana	**89.82 $\pm$ 0.66**	89.48 $\pm$ 0.65	89.61 $\pm$ 0.93
Breast Cancer	**95.83 $\pm$ 1.90**	95.16 $\pm$ 1.61	95.33 $\pm$ 2.04
Diabetes	**76.40 $\pm$ 1.40**	73.40 $\pm$ 3.14	75.33 $\pm$ 3.14
Fourclass	**98.41 $\pm$ 1.14**	96.23 $\pm$ 1.98	97.94 $\pm$ 1.18
German Numer	**74.75 $\pm$ 2.96**	68.45 $\pm$ 3.55	70.50 $\pm$ 4.24
Phishing	**92.75 $\pm$ 0.61**	89.90 $\pm$ 1.19	86.00 $\pm$ 2.49
Splice	**83.88 $\pm$ 1.24**	50.05 $\pm$ 5.72	51.20 $\pm$ 5.85
Svmguide1	**96.55 $\pm$ 0.37**	95.78 $\pm$ 0.83	96.05 $\pm$ 0.60
Svmguide3	**79.28 $\pm$ 3.10**	72.92 $\pm$ 2.70	73.48 $\pm$ 2.28

5.2 Application to smart steering in wireless mesh networks

In this section, we test the presented approach of Fourier feature learning on the steering data that is collected from a Wi-fi mesh network, cf. [8] for the details. Wi-fi mesh networks are widely used to achieve a better coverage of the area being served and thus provide an enhanced internet connection to the clients (devices that are connected to internet). As illustrated in Figure 3, a wi-fi mesh network has multiple access points (APs) such that a client (which can be possibly moving) is typically connected to internet through the AP that best serves. Moreover, each AP typically contains two frequency bands (interfaces), 2.4 GHz and 5 GHz, that the clients can use to access to internet. Therefore, in an attempt to receive a higher quality of internet connection, clients can not only switch from one AP to another but also from one interface to another. From the perspective of an authorized central agent in this setting, it is important by exploiting the mesh structure to optimally manage and distribute the clients to the available APs such that the overall quality of the internet connection among the clients of the network is maximized. To this end, the metrics such as total cost per link (summation of the cost of each connection from a client to gateway AP) and received signal strength indicator (RSSI) play a crucial role in measuring the connection quality that is to be maximized, as explained in [8].

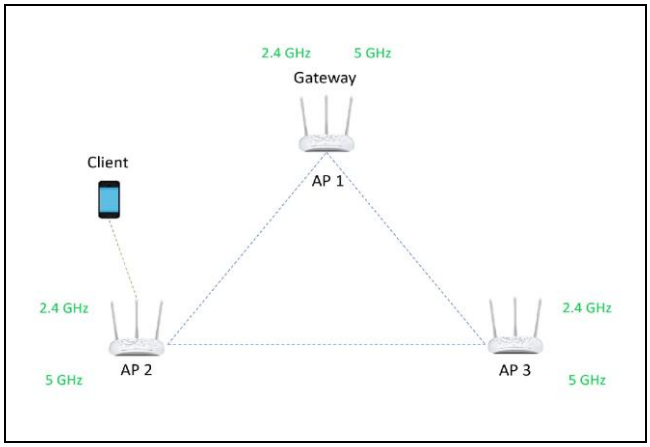

Figure 3. A Wi-Fi mesh network is illustrated. In this example, the network includes three access points (APs) with two interfaces (2.4 GHz and 5 GHz) per each, and we observe a client that is connected to the second AP.

A “steering action" refers to the request of the authorized central agent to guide a client to switch from the current AP or interface (that it is currently connected) to another AP or interface. Possible actions are: steering a client between different interfaces within the same AP, between different APs within the same interface, or between different interfaces of different APs. However, the outcome is not always successful, namely, the requested switch is not always realized and may eventually fail due to several reasons such as inaccurate reporting of the metrics or a vendor related issue. In this case of persistent un-switching (despite multiple steering actions), the corresponding client is known as the ``sticky client" which causes the inefficient use of resources, i.e., APs, compromising the connection quality [8].

Among possible steering strategies, we emphasize that the approach of issuing steering actions based on a predetermined rule is certainly not optimal as it does not learn from previous actions and outcomes. On the other hand, understanding when exactly and under what conditions of the aforementioned metrics the steering actions succeed or fail has the potential to help the agent issue steering actions more successfully. Namely, learning from previous steering actions and the corresponding

outcomes based on, for instance, a machine learning approach in a data driven and smart manner, the issued actions would be more likely to succeed. This would not only enhance the quality of internet connection, but also decrease the overhead of constantly communicating failing actions to a client that is perhaps sticky.

To this end, in [8], a binary classification problem is defined in which the ``steering data" describe steering actions by the aforementioned metrics, i.e., features, (signal strength and a certain link cost regarding the current connection as well as the targeted one: Current RSSI, Current Cost, Target RSSI and Target Cost) at the moment of the action as well as the corresponding outcome, i.e., labels, as successful or unsuccessful. We test the TL algorithm and compare it with the algorithm presented in [8] (which is essentially the SL algorithm) based on this steering data and present our average classification results in Figure 4 (over 1000 trials each of which is a random permutation of the steering data of [8]) with respect to the amount of data as streamed in an online application. For a fair comparison, we follow the same experiments conducted in [8] based on the feature pairs Target RSSI-Current RSSI and Current Cost-Current RSSI for switchings from 5 Ghz to 2.4 GHz.

We also include results of a linear perceptron (no kernel transformation), named as “No transform" in the figure. We first observe that the algorithms TL and SL strongly and uniformly outperform the linear perceptron in both cases of feature pairs. This shows that the steering is a nonlinear problem. Moreover, the TL algorithm strongly and uniformly outperforms the SL algorithm, indicating the high efficacy of the Fourier feature learning in the case of the steering data. Finally, in Table 5, and based on all 6 feature pairs, we evaluate the performance of the TL algorithm, and observe that the TL algorithm strongly outperforms the previously published results (given in parentheses as SL algorithm) uniformly in all cases. This, once again, reinforces that the presented approach of Fourier feature learning is superior over the perceptron in the kernel space and yields significantly superior steering results over the prior literature.

6 Conclusion and future work

We presented a single hidden layer feedforward neural network (SLFN) for Fourier feature learning, and investigated various training strategies: 1) single layer learning that yields a perceptron in the kernel space, 2) forward Fourier feature selection used with the single layer learning, 3) SLFN trained with stochastic gradient descent optimization, which is named as the TL algorithm, and 4) SLFN with coordinate descent optimization. The learned Fourier features were evaluated in terms of nonlinear classification based on 10 benchmark datasets from various fields such as bioinformatics and diagnostics. We observed that the TL algorithm (with the cross entropy loss) strongly outperforms the other strategies. Moreover, the TL algorithm performs comparable with the SVM with rbf kernel, which indicates that Fourier feature learning powerfully compensates for the strong max margin concept of SVM.

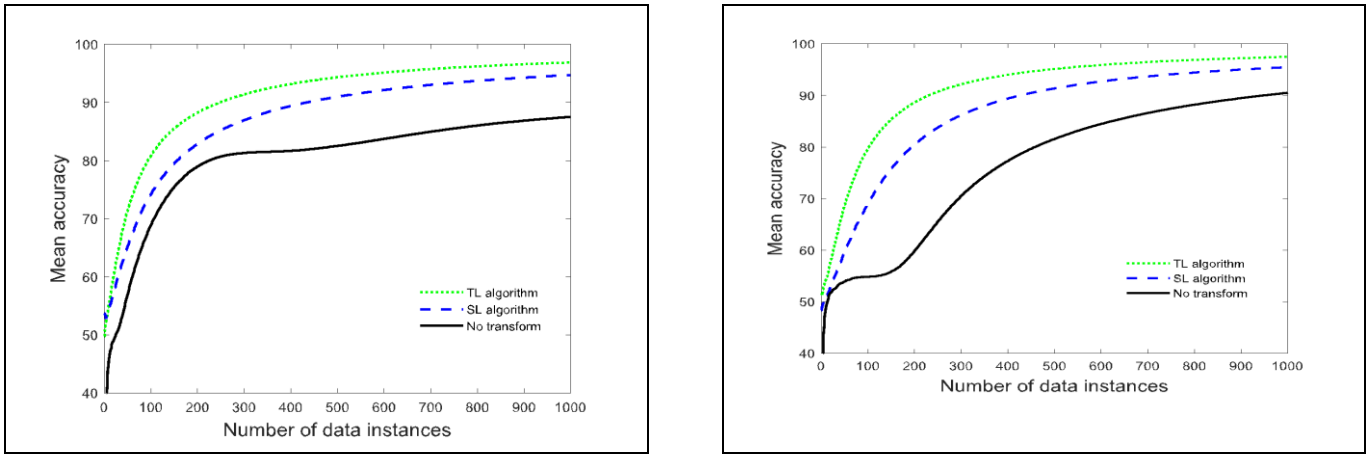

Figure 4. Comparisons of the TL algorithm with the results of [8] (SL algorithm) and with linear perceptron (“No transform") with the feature pairs Target RSSI-Current RSSI (left) and Current Cost-Current RSSI (right).

Table 5. Comparison of the TL algorithm (upper) with the results of [8] (SL algorithm lower in parenthesis).

	Multi AP			
	From 2.4 GHz to 5 GHz		From 5 GHz to 2.4 GHz	
	Same AP	Different AP	Same AP	Different AP
Current Cost - Target RSSI	**96.83**	**90.83**	**93.00**	**93.50**
	(94.94)	(87.71)	(91.90)	(89.52)
Target Cost - Target RSSI	**98.50**	**95.16**	**97.37**	**99.00**
	(94.28)	(88.06)	(95.75)	(95.75)
Target RSSI - Current RSSI	**96.00**	**97.00**	**98.87**	**99.00**
	(94.26)	(93.40)	(95.42)	(95.04)
Current RSSI - Target Cost	**95.00**	**89.33**	**98.00**	**99.50**
	(92.19)	(85.27)	(94.34)	(98.08)
Current Cost - Current RSSI	**97.00**	**97.16**	**98.87**	**99.16**
	(94.35)	(91.86)	(94.75)	(97.99)
Target Cost - Current Cost	**74.16**	**74.16**	**94.50**	**98.33**
	(60.67)	(57.71)	(66.38)	(53.34)

On the other hand, the TL algorithm is computation as well as space wise highly efficient, whereas SVM is prohibitively complex. In addition, we demonstrated an application of the TL algorithm to smart steering in wireless mesh networks, and tested with the steering data from a previous study. In this application, the task is to classify a steering action (i.e. a request from a network client to switch from one connection to another) as successful or not. Hence, a better connectivity among all network clients can be realized overall. The TL algorithm has been again observed to be significantly superior over the previously reported results as it strongly outperforms the perceptron trained in the kernel space.

As future work, we plan to compare the investigated approach of Fourier feature learning with the alternative approach of learning of the kernel parameter. Since the Fourier features are directly a function of the kernel parameter through sampling; in a neural network set-up, one can directly backpropagate to the parameter to estimate its value based on a classification loss (after an appropriate reparameterization). In this alternative, the advantage is that the parameter complexity is small because the kernel parameter is only a scalar (for instance, the bandwidth of the rbf kernel). Whereas the learning of the Fourier features requires the estimation of, $D\ x\ (d+1)$ many, far more parameters (with the rbf kernel), with D being the number of Fourier features and d being the dimension. Despite its advantage of reduced parameter complexity, learning of the kernel parameter restricts the data modeling capability to the specific kernel being used, which might be seen as a disadvantage. Hence, a comparison to understand this trade-off would be useful. Another important future research direction is to fit an unisotropic Gaussian kernel to the data through the learning of the Fourier features. Note that the sampling distribution of the features is the inverse Fourier transform of the kernel. Then, one can apply the forward Fourier transform to the sample distribution of the features when the training (Fourier feature learning) converges. Here, the forward transform can model an unisotrotopic Gaussian kernel for the data if the sample distribution of the features is Gaussian. As a result, one can constrain the learning of the Fourier features and ensure Gaussianity by, for example, using the Kullback-Leibler divergence (in the loss function) between the sample distribution and a Gauss distribution with zero mean and sample covariance (at each step in training). The resulting unisotropic Gauss kernel has the potential to provide useful insights to modelling the manifold that the data lie in.

7 Acknowledgements

This work was supported by The Scientific and Technological Research Council (TUBITAK) of Turkey under Contract 118E268.

8 Author contribution statement

This work was completed as a part of the MS thesis of the first author Bulut KUŞKONMAZ, where the second author Hüseyin ÖZKAN was the thesis advisor.

9 Ethics committee approval and conflict of interest statement

There is no need to obtain permission from the ethics committee for the article prepared.

There is no conflict of interest with any person / institution in the article prepared.

10 References

[1] Hofmann T, Schölkopf B, Smola AJ. "Kernel methods in machine learning". *The Annals of Statistics*, 36(3), 1171-1220, 2008.

[2] Cortes C, Vapnik V. "Support-vector networks". *Machine Learning*, 20(3), 273-297, 1995.

[3] Scholkopf B, Sung KK, Burges CJ, Girosi F, Niyogi P, Poggio T, Vapnik V. "Comparing support vector machines with gaussian kernels to RBF classifiers". *IEEE Transactions on Signal Processing*, 45(11), 2758-2765, 1997.

[4] Jaakkola TS, Haussler D. "Probabilistic kernel regression models". *Artificial Intelligence and Statistics*, Ft. Lauderdale, FL, USA, 3-6 January 1999.

[5] Kerpicci M, Ozkan H, Kozat SS. "Online anomaly detection with bandwidth optimized hierarchical kernel density estimators". *IEEE Transactions on Neural Networks and Learning Systems*, 32(9), 4253-4266, 2020.

[6] Lanckriet GR, Cristianini N, Bartlett P, Ghaoui LE, Jordan MI. "Learning the kernel matrix with semidefinite programming". *Journal of Machine Learning Research*, 5(1), 27-72, 2004.

[7] Rahimi A, Recht B. "Random features for large-scale kernel machines". *Neural Information Processing Systems,* Vancouver, B.C., Canada, 3-6 December 2007.

[8] Kuskonmaz B, Ozkan H, Gurbuz O. "Machine learning-based smart steering for wireless mesh networks". *Ad Hoc Networks*, 88(1), 98-111, 2019.

[9] Lu J, Hoi SC, Wang J, Zhao P, Liu ZY. "Large scale online kernel learning". *Journal of Machine Learning Research*, 17(1), 1613-1655, 2016.

[10] Băzăvan EG, Li F, Sminchisescu C. "Fourier kernel learning". *European Conference on Computer Vision,* Berlin, Germany, 7-13 October 2012.

[11] Nguyen T, Le T, Bui H, Phung D. "Large-scale online kernel learning with random feature reparameterization". *International Joint Conferences on Artificial Intelligence*, Melbourne, Australia, 19-25 August 2017.

[12] Oliva JB, Dubey A, Wilson AG, Póczos B, Schneider J, Xing EP. "Bayesian nonparametric kernel-learning". *Artificial Intelligence and Statistics,* Cadiz, Spain, 9-11 May 2016.

[13] Li CL, Chang WC, Mroueh Y, Yang Y, Póczos B. "Implicit kernel learning". *Artificial Intelligence and Statistics*, Naha, Okinawa, Japan, 16-18 April 2019.

[14] Wang A, Law L, Miscouridou X, Mider M, Ip S. "Kernel learning via random fourier representations". Warwick Statistics Programme, Oxford University, Oxford, England, Reports and Presentations, 2016.

[15] Xie J, Liu F, Wang K, Huang X. "Deep kernel learning via random fourier features". *arXiv Preprint*, 2019. https://doi.org/10.48550/arXiv.1910.02660.

[16] Xue H, Wu ZF, Sun WX. "Deep spectral kernel learning". *International Joint Conferences on Artificial Intelligence*, Macao, China, 10-16 August 2019.

[17] Can B, Ozkan H. "A neural network approach for online nonlinear neyman-pearson classification". *IEEE Access*, 8(1), 210234-210250, 2020.

[18] Porikli F, Ozkan H. "Data driven frequency mapping for computationally scalable object detection". *IEEE International Conference on Advanced Video and Signal Based Surveillance*, Klagenfurt, Austria, 30 August–2 September 2011.

[19] Alpaydin E. *Introduction to Machine Learning*. 4th ed. Cambridge, MA, USA, MIT Press, 2020.
[20] Muller KR, Mika S, Ratsch G, Tsuda K, Scholkopf B. "An Introduction to kernel-based learning algorithms". *IEEE Transactions on Neural Networks*, 12(2), 181-201, 2001.
[21] Rudin W. *Fourier Analysis on Groups*. New York, USA, Wiley, 2011.
[22] McClellan JH, Schafer RW, Yoder MA. *Signal Processing First*. 1st ed. Upper Saddle River, NJ, USA, Pearson, 2003.
[23] Burges CJ. "A tutorial on support vector machines for pattern recognition". *Data Mining and Knowledge Discovery*, 2(2), 121-167, 1998.
[24] Wright S. "Coordinate descent algorithms". *Mathematical Programming*, 151(1), 3-34, 2015.
[25] Chang CC, Lin CJ. "LIBSVM: A library for support vector machines". *ACM Transactions on Intelligent Systems and Technology*, 2(3), 1-27, 2011.