

**INVESTIGATION OF FOURIER FEATURES IN NEURAL
NETWORKS AND AN APPLICATION TO STEERING IN MESH
NETWORKS**

by
BULUT KUŞKONMAZ

Submitted to the Graduate School of Social Sciences
in partial fulfilment of
the requirements for the degree of Master of Electronics Engineering

Sabancı University
September 2020

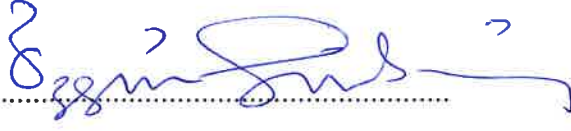
INVESTIGATION OF FOURIER FEATURES IN NEURAL NETWORKS
AND AN APPLICATION TO STEERING IN MESH NETWORKS

APPROVED BY:

Assist. Prof. Dr. HÜSEYİN ÖZKAN
(Thesis Advisor)



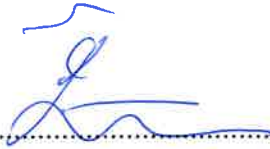
Prof. Dr. ÖZGÜR GÜRBÜZ
(Thesis Co-advisor)



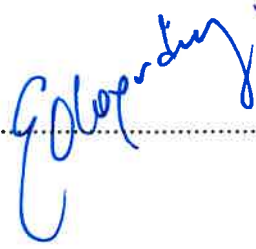
Prof. Dr. ALBERT LEVİ



Assist. Prof. Dr. ÖZNUR TAŞTAN



Assist. Prof. Dr. ERDEM AKAGÜNDÜZ



DATE OF APPROVAL: 03 September 2020

THESIS AUTHOR 2020 ©

All Rights Reserved

ABSTRACT

INVESTIGATION OF FOURIER FEATURES IN NEURAL NETWORKS AND AN APPLICATION TO STEERING IN MESH NETWORKS

BULUT KUŞKONMAZ

Electronics Engineering M.Sc. Thesis, September 2020

Thesis Advisor: Assist. Prof. Dr. Hüseyin Özkan

Co-advisor: Prof. Dr. Özgür Gürbüz

Keywords: Fourier features, Neural networks, SLFN, Classification, Kernel,
Steering, Mesh networks

Random Fourier features provide one of the most prominent ways to classify large-scale data sets when the classification is nonlinear. However, Fourier features, in its original proposal, are randomly drawn from a certain distribution and are not optimized. In this thesis, we investigate the use of Fourier features by a single hidden layer feedforward neural network (SLFN) and optimize those features (instead of drawing randomly) with several gradient-descent based approaches. The optimized 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. The resulting classification accuracy is compared with the results of SVM with RBF kernel. Particularly, (1) we tune the parameters such as the hidden layer size and RBF kernel bandwidth, and (2) test with ten different classification data sets. The introduced SLFN provides substantial computational gains with similar accuracy figures compared to the ones of SVM. We also test our SLFN for steering in wireless mesh networks and observe promising smart steering capabilities.

ÖZET

FOURIER ÖZİNİTELİKLERİNİN SINIR AĞLARI İLE İNCELENMESİ VE ÖRGÜ AĞLARDA BAĞLANTI YÖNLENDİRMEYE UYGULANMASI

BULUT KUŞKONMAZ

ELEKTRONİK MÜHENDİSLİĞİ YÜKSEK LİSANS TEZİ, EYLÜL 2020

Tez Danışmanı: Assist. Prof. Dr. Hüseyin Özkan

İkinci Danışman: Prof. Dr. Özgür Gürbüz

Anahtar Kelimeler: Fourier öznitelikleri, Sinir ağları, SLFN, Sınıflandırma,
Çekirdek, Bağlantı yönlendirme, Örgü ağları

Rastgele Fourier öznitelikleri, sınıflandırma doğrusal olmadığında büyük ölçekli veri kümelerini sınıflandırmanın en belirgin yollarından birini sağlar. Bununla birlikte, orjinal önerisinde Fourier öznitelikleri, belirli bir dağıtımdan rastgele çekilir ve optimize edilmez. Bu tezde, Fourier özniteliklerinin tek gizli katmanlı ileri beslemeli sinir ağı (SLFN) ile kullanımını araştırıyor ve bu öznitelikleri (rastgele seçim yerine) çeşitli gradyan-inişi tabanlı yaklaşımlarla optimize ediyoruz. Optimize edilmiş Fourier öznitelikleri, radyal bazlı fonksiyondan (RBF çekirdeği) çıkarılır ve çıkış katmanının takip ettiği SLFN'nin gizli katmanında uygulanır. Ortaya çıkan sınıflandırma doğruluğu, RBF çekirdeği ile SVM'nin sonuçlarıyla karşılaştırılır. Özellikle, (1) gizli katman boyutu ve RBF çekirdek bant genişliği gibi parametreleri ayarlıyoruz ve (2) on farklı sınıflandırma veri seti ile test ediyoruz. Sunulan SLFN, SVM'ye kıyasla benzer doğruluk rakamlarına sahip önemli hesaplama kazançları sağlar. Ayrıca kablosuz ağ ağlarında bağlantı yönlendirme için SLFN'mizi test ediyor ve gelecek vaat eden akıllı bağlantı yönlendirme kabiliyetlerini gözlemliyoruz.

ACKNOWLEDGEMENTS

I would like to dedicate my sincere appreciation to my advisor Assist. Prof. Dr. Hüseyin Özkan for his endless support and brilliant suggestions to complete this thesis. I gained the ability to be an engineer thanks to him. I also would like to thank my co-advisor, Prof. Dr. Özgür Gürbüz, for her immense help in both academic life and civil life. She has always been there to give me the right advice whenever I needed it since my undergraduate education.

I would like to thank Prof. Dr. Albert Levi, Assist. Prof. Dr. Öznur Taştan, and Assist. Prof. Dr. Erdem Akagündüz for their meticulous evaluation of my thesis.

I am grateful to my family, İbrahim, Ayten, and Güneş for their love and support throughout my whole education.

I am grateful to my lab mates, Ali, Kutay, Mehmet, and Sandra who make my graduate life at Sabancı University easier and more enjoyable. Special thanks go to my 'Japanese Friends', who gave me true friendship and love. I also want to thank Berke, who started his university journey with me as a roommate and continued as one of my true friends. Thanks to my close friend Oğuz, whom I studied Electronics Engineering with. I want to thank Volkan for being an amazing roommate and true friend who I can really trust. Finally, heartfelt thanks to the people of Kocaeli Doğa Sporları Kulübü (KODOSK), for making me proud of being a member of KODOSK.

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

Dedicated to my family and friends

TABLE OF CONTENTS

LIST OF TABLES	x
LIST OF FIGURES	xi
1. INTRODUCTION	1
1.1. Thesis Contributions.....	4
1.2. Thesis Organization	5
2. RELATED WORK	6
3. LEARNING OF FOURIER FEATURES WITH A NEURAL NET- WORK	11
3.1. Random Kernel Expansion	13
3.2. Learning Fourier Features	16
3.3. Various Training Approaches for Learning Fourier Features.....	18
3.3.1. Single Layer Learning (SL).....	18
3.3.2. Fourier Feature Selection (FFS).....	18
3.3.3. Two Layer Learning (TL)	19
3.3.4. Batch-Based Two Layer Learning (TL-B).....	19
3.3.5. Epoch-Based Two Layer Learning (TL-E)	19
3.4. Experiments	19
4. AN APPLICATION OF THE PROPOSED APPROACH: SMART STEERING FOR WIRELESS MESH NETWORKS	25
4.1. Introduction	25
4.1.1. Related Work	28
4.1.2. Chapter Organization	29
4.2. Problem Description	29
4.3. The Proposed Classification Approach for Smart Steering	31
4.3.1. Classification Analysis in the Batch Setting	33
4.3.2. Online Classification for Real-time Smart Steering	35

4.3.2.1. Perceptron in the Randomized Kernel Space: Online Kernel Perceptron	36
4.4. Experimental Results	38
4.4.1. Steering Data	39
4.4.2. Results of the Batch Analysis	40
4.4.3. Results of Online Classification	45
4.5. Discussion.....	49
5. CONCLUSION	51
BIBLIOGRAPHY.....	52
6. Bibliography.....	52

LIST OF TABLES

Table 3.1. Benchmark details as provided in [1]	20
Table 3.2. Cross validation results: average bandwidth parameter g (upper) and number D of units in the hidden layer (lower) with the corresponding standard deviations in each case	21
Table 3.3. Benchmark results of TL, SL, TL-E and TL-B algorithms with CE/MSE loss and minibatch/SGD optimizers on ten different data sets	22
Table 3.4. Comparison of TL algorithm (CE and mini batch) with SVM (rbf kernel) in terms of classification accuracy	23
Table 3.5. Comparison of two layer learning approach, single layer learning approach, single layer learning with chosen Fourier features (FFS) and linear SVM with Fourier features in terms of the mean and standard deviation of accuracy	24
Table 4.1. Accuracy results of nonlinear SVM in single AP scenario.	41
Table 4.2. Accuracy results of nonlinear SVM in the multi AP scenario. ..	44
Table 4.3. Error rate results of online kernel perceptron in the multi AP scenario.	48
Table 4.4. Accuracy results of TL learning algorithm in the multi access point (AP) scenario	49

LIST OF FIGURES

Figure 3.1. An example of linear classification in (a) and nonlinear classification in (b)	11
Figure 3.2. Visual interpretation of kernel trick that allows nonlinear classification with linear techniques	12
Figure 3.3. Visual representation of our single hidden layer feedforward neural network (SLFN), where d is dimension of the input data instance x and D is the size of the hidden layer implementing the mapping with random Fourier features. The hidden layer activation is sinusoidal producing the Fourier features $\{\cos(w_r^T x_t + b_r)\}_{r=1}^D$. Initially, $(w_r, b_r)_{r=1}^D$ are randomly drawn from the density $N(0, 2gI) \times U(0, 2\pi)$ which is guaranteed to provide -even initially- powerful nonlinear classification as it implements the radial basis function (rbf) kernel. Subsequently, this powerfully initialized SLFN learns Fourier features as its hidden layer activations via the backpropagation based optimization.	16
Figure 4.1. An example home mesh network.	26
Figure 4.2. Wi-Fi wireless mesh networks with Batch ML and online ML for smart steering.	32
Figure 4.3. The boundary evolution of the banana dataset. In (a), SVM classification with 1000 data instances is presented (error penalty parameter is 8 and kernel bandwidth parameter is 0.7). In (b), (c) and (d), online kernel perceptron classification is presented with 100, 250 and 1000 data instances, respectively.	37
Figure 4.4. Logged data from a typical house use for a single AP (a specific window).	39

Figure 4.5. (a) Linear SVM classification based on Current Cost and Current RSSI for clients in single AP scenario with transition from 2.4 GHz to 5 GHz and (b) nonlinear SVM classification based on Current Cost and Current RSSI for clients in single AP scenario with transition from 2.4 GHz to 5 GHz.	40
Figure 4.6. Classification results of various feature pairs for steering a client from 2.4 GHz interface of an AP to 5 GHz interface of different AP's.	42
Figure 4.7. Classification results with respect to the (Current RSSI, Target RSSI) pair for steering a client (a) from 2.4 GHz to 5 GHz interface of different AP's (b) from 5 GHz to 2.4 GHz interface of different AP's (c) from 2.4 GHz interface of an AP to 5 GHz interface of the same AP (d) from 5 GHz interface of an AP to 2.4 GHz interface of the same AP.	43
Figure 4.8. The boundary evolution of the dataset of the transition from 2.4 GHz to 5 GHz of different AP's with the feature pair (Target RSSI, Current RSSI). In (a), SVM classification with 1000 data instances is presented (error penalty parameter is 2 and kernel bandwidth parameter is 0.8). In (b), (c) and (d), online kernel perceptron classification is presented with 300, 600 and 1000 data instances (kernel bandwidth parameter is 0.8), respectively.	46
Figure 4.9. Online kernel perceptron classification results based on the feature pairs (Target RSSI, Current RSSI) and (Current Cost, Current RSSI) for clients in multi AP scenario with transition from 2.4 GHz to 5 GHz are given in (a) and (b), respectively, up to $D = 10$	47

1. INTRODUCTION

Classification is an important problem in machine learning and binary classification is a fundamental type [2]. Linear classification is a commonly used approach but it is insufficient when the data is not linearly separable. In such cases, nonlinear classification is a solution, e.g., kernel machines. Approximation of classification functions of nonlinear kernel machines with Random Fourier Features (RFF) is one of the viable techniques [3] since it allows large scale nonlinear classification with linear techniques (after the randomized kernel expansion) in a computationally efficient manner [3, 4, 5, 6].

For example, Support Vector Machines (SVM) [2] is a supervised machine learning algorithm that generates a binary classifier based on labeled data $(X, Y) \in R^{N \times d} \times \{-1, 1\}^{N \times 1}$, where d is the dimension, N is the number of data instances, X is the data matrix of features that are also associated with binary labels Y describing the class memberships. SVM learns a separating hyperplane in the feature space. The hyperplane is defined by its normal vector $\alpha \in R^{d \times 1}$ and a bias $\beta \in R$ such that the resulting classifier is in the form of $f(x) = \text{sign}(\alpha^T x + \beta)$. Then, SVM optimizes the hyperplane parameters (α, β) using a convex quadratic programming [2].

If one aims to separate classes in nonlinear fashion with SVM, kernelization [2, 7] is the suitable solution. Instead of relying on dot products in defining the similarity between two instances x_i and x_j , one typically uses a kernel (satisfying Mercer's condition), e.g., we use radial basis function (RBF) kernel $k(x, y) = e^{-g\|x-y\|^2}$ in this thesis, to re-define the similarity (g is the bandwidth parameter that leads to more complex or simpler nonlinear models when relatively higher or smaller values are set). This process transforms the data into relatively higher dimension in which the data becomes linearly separable although it requires nonlinear classification in its original observation space [2].

An alternative way to perform nonlinear classification is random kernel expansion via random Fourier Features [3]. For any two instances $x_i, x_j \in R^{d \times 1}$, one produces the corresponding pair of transformed instances $x_i, x_j \rightarrow z_i, z_j \in R^{2D \times 1}$ such that

$k(x_i, x_j)$ can be approximated arbitrarily well by $z_i^T z_j$ as D (dimension of the transform) increases, i.e., $k(x_i, x_j) \simeq z_i^T z_j$. The approximation $k(x_i, x_j) \simeq z_i^T z_j$ is based on projections by random Fourier features as an application of the Bochner's theorem [3]. Bochner's theorem allows us to draw projection vector w from $p(w)$ with $p(w) = N(w; \mu, \Sigma)$ with zero mean $\mu = 0$ and covariance $\Sigma = 2gI$ (I is the identity matrix of the appropriate size). Having the kernel space explicitly and compactly constructed via the transform

$$R^{d \times 1} \ni x_t \rightarrow z_t = \frac{1}{\sqrt{D}} [r_{w_1}(x_t), r_{w_2}(x_t), \dots, r_{w_{D-1}}(x_t), r_{w_D}(x_t)]^T \in R^{2D \times 1}$$

for each instance for a given data stream $\{x_t\}$, where $r_w(x) = [\cos(w^T x), \sin(w^T x)]$, then a linear classifier in the z -space can solve any nonlinear classification problem provided that the kernel is chosen suitably, e.g., RBF kernel with an appropriate kernel parameter g . We use this form of transformation in steering application where we discuss in Chapter 4. There is an alternative way to achieve this transformation. w is basically the random projection direction that enable us to use the mapping $Z_w(x) = \sqrt{2} \cos(w^T x + b)$,

$$R^{d \times 1} \ni x_t \rightarrow z_t = \frac{1}{\sqrt{D}} [Z_{w_1}(x_t), Z_{w_2}(x_t), \dots, Z_{w_{D-1}}(x_t), Z_{w_D}(x_t)]^T \in R^{D \times 1}$$

(where b is chosen from uniformly from $[0, 2\pi]$) provide random Fourier features $Z_w(x)$ [3]. Since there exists $\alpha \in R^D$ corresponding to any $\bar{\alpha} \in R^{2D}$ such that $\bar{z}^T \bar{\alpha} = z^T \alpha$ which can be straightforwardly observed by phasor addition [8] We use this version of transformation in Chapter 3 where we investigate Fourier features via single hidden layer feedforward neural network (SLFN).

Comparison of the computational complexity of SVM with RBF kernel and on-line linear classification with random Fourier features (resulting online nonlinear classification with original features) provides an important observation about the practicality of the random Fourier features in the context of large scale online data processing. SVM, as an algorithm defined in the batch setting, has the computational complexity for training and testing $O(N^3)$ and $O(NN_{test})$, respectively. This amount of computational complexity is highly challenging with the large scale data in real-time applications. On the other hand, random Fourier features yield online algorithms with constant $O(1)$ online processing complexity (in total $O(N)$ after processing N data instances) at the cost of a transformation complexity $O(Dd)$, both are per instance, and D and d are relatively small compared to N or N_{test} .

For example, considering the test scenario of SVM, classification of a test instance x requires computing $f(x) = \sum_{i=1}^{N_{SV}} \alpha_i k(x, x_i) + \beta$. This requires N computation in

the worst case with $O(Nd)$ complexity since the number N_{SV} of support vectors can be as large as the number N of training instances. A situation like this would be inefficient in processing large data sets. Hence, approximating the function $f(x) \simeq \sum_{i=1}^N \alpha_i z_x^T z_{x_i} + \beta = z_x^T (\sum_{i=1}^N \alpha_i z_{x_i}) + \beta$ reveals that the SVM kernel classification is a linear separator after the transformation with random Fourier features, which is only a constant $O(1)$ online processing complexity (in total $O(N)$ after processing N data instances) per instance.

However, it is possible to have better classification by optimizing the projection vectors w that are initially randomly drawn from $p(w)$. Although random Fourier features have been used with great success in especially large scale classification qualifying Fourier features as good features, the fact that they are mostly randomly used and the possibility that they can be learned without completely relying on data independent randomization is a promising research direction. To this end, several studies have been conducted to learn Fourier features, with some being fairly recent as simultaneous developments with the work in the presented thesis. For example, the authors optimize the Fourier features using scalable optimization methods for learning kernels and conduct experiments on a couple of image data sets in [9]. Unlike the study in [9], we optimize the Fourier features using SLFN and hold comprehensive experiments using different classification data sets. In [4], they reparameterize the random features by lifting the source of randomness to another space to optimize them using stochastic gradient descent and keep the original kernel parameters untouched, whereas we directly update the kernel parameters via SLFN with no need to preserve the original kernel parameters. The authors in [10] implement the random Fourier feature method using neural networks which contain multiple layers for learning kernels with small and large scale data sets. This network they propose can optimize multiple kernel parameters using backpropagation and they conduct experiments on several image and classification data sets. On the other hand, we propose an SLFN that exploits Fourier features to optimize the kernel, provide several methods using backpropagation, and hold experiments that are more comprehensive compared to the study in [10]. Unlike the studies we discuss until here, we deploy our SLFN to steering data set in a wireless mesh network and obtain outstanding results.

In this thesis, we investigate the use of Fourier features in neural networks, such that both the Fourier features as well as linear classifier thereafter are learned in the context of nonconvex neural network training. Note that, in a similar fashion with the aforementioned related examples in the literature, stochastic gradient based training can be chosen for real-time scalability to voluminous data. Other optimizers such as minibatch approaches with nesterov momentum updates can surely be used as well.

In particular, we propose a SLFN which transforms data into high dimension at the first and hidden layer and then transformed data passes through the second and output layer which performs linear classification. The hidden layer of the proposed SLFN implements Fourier features as the hidden layer activations, which reposes the data in a linearly separable manner (even if it is originally nonlinear) in the high dimension enabling the use of a linear classifier afterwards. Hence, the training of the introduced SLFN does not only learn a linear classifier in the high dimensional transform space as a nonlinear classifier in the original space, but also optimizes the Fourier features in a data driven manner getting rid of the data-independent randomness of Fourier features in its original proposal [3]. To this end, we analyze and compare four different gradient descent based training approaches on an extensive benchmark of datasets, and finally demonstrate a real life application of smart steering in wireless mesh networks.

1.1 Thesis Contributions

Main contributions of the presented thesis can be summarized as follows.

- We investigate Fourier features in neural networks in detail and deploy random Fourier features via SLFN to perform nonlinear classification, while tuning the parameters of hidden layer unit size D and bandwidth parameter g for efficiency. We apply several training strategies for this investigation.
- We learn the optimal projection directions for Fourier features, which are drawn from a certain (Fourier transform the kernel in hand to be more precise, which is a proper distribution) distribution at the initialization of the introduced SLFN, using gradient descent-based backpropagation algorithms. We compare our proposed learning algorithms with SVM.
- We apply our SLFN to the steering data set as an application in wireless mesh networks. We conduct a batch analysis via SVM for characterizing the steering data, and based on this analysis, we propose a batch technique for smart steering approach.
- Based on the findings of our batch analysis, we develop an online learning approach (online kernel perceptron), namely, an online technique for smart steering which is a data-driven, adaptive, real-time algorithm applied to steer-

ing for the first time in the literature.

1.2 Thesis Organization

The rest of the thesis is organized as follows. In Chapter 2, we provide the background of the thesis and related work. Chapter 3 presents the SLFN and algorithms we propose. Results of our proposed SLFN are given in Chapter 3 as well. In Chapter 4, we introduce our classification based approach for smart steering in wireless mesh networks, including batch and online steering algorithms. Experiments in Chapter 4 present the steering data we use, preprocessing and our end results. Chapter 5 draws our conclusions.

2. RELATED WORK

In this chapter, we provide the related work in comparison to the presented thesis. Various prominent studies are discussed which use random Fourier features (RFF). Studies that are related to the optimization of Fourier features for several tasks are discussed as well. We also discuss the use of a single hidden layer feedforward neural network (SLFN) for various purposes

This thesis focuses on the investigation of RFF using SLFN and optimizes Fourier features via SLFN to address binary classification problem. A large amount of research has conducted on random Fourier features (RFF), which is a kernel method has proposed in [3]. The authors in [11] improve the kernel method in terms of variance of Gaussian kernel and approximation error. In [12], the authors discuss a detailed investigation on the approximation quality of RFF and propose a proper RFF approximation for the derivatives of a kernel function. Authors suggest applying linear clustering algorithms after mapping the data points to a low-dimensional feature space in [13]. The use of gradient descent for optimizing Fourier features and the Nyström method to approximate kernel functions are introduced in [5]. These studies give an investigation of RFF for its novelty, approximation, optimization, and combine with other algorithms. In this thesis, we investigate RFF using SLFN and deploy our SLFN to steering application in a wireless mesh network.

RFF has used in various ways to solve nonlinear machine learning problems, and this is one of the main focuses of this thesis. RFF based learning method is extended to multiple kernel learning for different channels or features by performing optimization in Fourier domain [9]. In [4], the authors address issues such as the computational complexity caused by a huge amount of data points and the problem of learning kernel parameter address by introducing the reparameterization. The authors propose Random Fourier features neural networks (RFFNet) in [10], which can approximate kernels on different layers using backpropagation for training. Unlike these authors, we focus on learning a single kernel and conduct more extensive experiment with our SLFN. The authors propose an efficient algorithm for large data sets in [14], in which variables of random Fourier features are sampled with a

Bayesian approach using a non-parametric mixture of Gaussians in BaNK algorithm. In [15], the authors introduce a generic kernel learning (IKL) which focuses on transforming a sample from a base distribution to another kernel to learn the sampling process of kernel distributions. Another method is the Nyström method, in which the authors calculate the low-rank matrix via data points to approximate the kernel matrix; they compare this method with the random Fourier feature method in [16]. The authors in [17] approximate a kernel by using one hidden layer neural network with RFF and optimize features with gradient descent. The authors introduce an algorithm in [6], which uses a sequence of Fourier features to provide the maximum realizable SVM classification margin for supervised learning. In [18], the authors treat the Fourier transform as a prior distribution over trigonometric functions and they introduce two PAC Bayesian-based methods. These random Fourier Features can also be optimized with an appropriate attitude, and this thesis concentrates on optimizing features and demonstrating an application to steering in wireless mesh networks.

Kernel methods can be applicable to different tasks. The authors apply the kernel convolutional layer which provides the kernel trick for the convolutional layers and they deploy it to small patches of the image in [19] to achieve a better classification result. Approximating multiple kernels can be useful to address classification problems. In [20], the authors apply kernel approximation relying on the insight of bag of words representation (BOW), and they propose efficient match kernels (EMK) to map local features to a low-dimensional feature space that enables linear classifiers. The authors in [21] extend the concept of Multiple Kernel Learning (MKL) to a kernel combination with regularization on the kernel parameters as general MKL (GMKL), which is a gradient descent based algorithm for binary classification. In [22], the authors propose multiple kernel learning (MKL) with group Lasso constraint on kernel weights that provides a proper classification for Alzheimer's disease. As discussed until here, kernel approximation can be successfully used to address various problems, and this thesis considers a kernel approximation (in particular, we comprehensively investigate random Fourier features in neural networks) and demonstrates an application to steering in wireless mesh networks.

RFF method becomes a popular concept in machine learning and researchers apply it in different ways. A generalized RBF kernel with a finite-dimensional approximate feature map, which is an algorithm that is independent of the number of support vectors, the authors introduce in [23]. Another study [24] to understand the sampling complexity of online kernel learning. In [25], the authors combine the concept of RFF with non-linear distributed networks where K nodes are connected and each node operates with its neighbors. In [26], the authors introduce the concept of

orthogonality into the RFF in which the Gaussian kernel matrix is replaced with a random orthogonal matrix; they call this method Orthogonal Random Features (ORF). The same authors further introduced the method Structured Orthogonal Random Features (SORF), in which discrete orthogonal matrices are used to have a fast matrix computation. The authors propose an algorithm called Correlated Nyström Views (XNV) in [27], which is a combination of the Nyström method and Canonical Correlation Analysis (CCA). In [28], they introduce locally compact abelian groups for random Fourier features. This new group, based on empirical observations on the exponentiated X^2 kernel, enables algorithms to build non-scale-invariant kernels and to approximate them linearly using RFF. So far, we discuss several studies that focus on combining RFF with different approaches and approximate kernels with various approaches rather than neural networks. Hence, we use SLFN to implement RFF and approximate kernel.

In this thesis, the neural network we consider for optimizing Fourier features is essentially an SLFN. The authors propose an incremental constructive method with different activation functions in [29] in which can be efficient to build an incremental feedforward network. In [30], authors discuss the classification regions that can be formed by SLFNs with bounded activation functions. They propose a new robust training algorithm for SLFNs in [31] and this algorithm uses linear nodes and tapped delay input for the network. Online sequential learning is an efficient learning approach that is particularly applicable to applications with real-time processing requirements. In [32], the authors propose online sequential learning based on Radial Basis Function (RBF) nodes for SLFNs to provide fast and accurate results. The authors propose Online Sequential Extreme Learning Machine (OS-ELM) in [33] which enables algorithms to perform sequential learning. OS-ELM can operate with both additive neurons and RBF kernels, and its parameters do not need tuning. These studies focus on solving problems using SLFN and apply different approaches via SLFN and we apply the RFF method using SLFN in this thesis.

Investigating the RFF using SLFN is one of the main focus of this thesis. The authors examine the behavior of the single and two hidden layer neural networks in [34] when backpropagation is used. Their experiments show that there is not much difference in single and two-layer networks in terms of trainability and classification accuracy. The same authors discuss that single-layer networks have slightly better trainability and classification accuracy results, and they show two-layer networks train easier when hidden layers have a more or less equal number of hidden units. In [35], the authors examine the hidden layers and activation functions and show how activation functions and gradients behave during training. As discussed so far, the investigation of SLFN can focus on trainability, classification accuracy, and the

state of the activation functions. Therefore, we investigate our RFF based SLFN in terms of classification accuracy and optimization of the parameters of RFF.

Extreme Learning Machine (ELM) is a fast and powerful learning technique for SLFNs that chooses input weights randomly and determines the output weights analytically [36] and this technique is similar to the work we do in this thesis. Various types of ELM are used in literature for various applications and problems such as in [37], a hybrid algorithm as a combination of ELM and differential evolution is proposed, called evolutionary extreme learning machine (E-ELM). The authors introduce a regularized ELM in [38] and this method works for noisy datasets as well. In [39], the authors propose optimally pruned ELM (OP-ELM), which eliminates unnecessary nodes and variables. In [40], the authors introduce the error minimized ELM (EM-ELM), which recursively updates its hidden layer and the number of hidden nodes until the desired error is calculated. The authors propose OS-ELM with forgetting mechanism (FOS-ELM) in [41] to address the timeliness problem that each data instance has a certain period for being valid. Despite the similarity of ELM and RFF in terms of using linear methods to optimize weights, a gradient-based approach is used to optimize Fourier features, unlike ELM, in this thesis.

The authors propose a fully complex extreme learning machine (C-ELM) in [42] for nonlinear channel equalization applications. The only difference between C-ELM and ELM is that it uses complex input weights and complex bias. [43] extends the ELM algorithm with radial basis function (RBF) networks for SLFNs and call it ELM-RBF. [44] proposes pruned ELM (P-ELM), which chooses the hidden nodes according to their contribution to the classification accuracy using probabilistic methods such as chi-squared and information gain. In [45], the authors introduce an algorithm that uses the essentials of the ELM algorithm combined with a classical cross-validation approach and they call it ensemble-based ELM (EN-ELM). MultiLayer Extreme Learning Machine (ML-ELM) is another type of ELM that [46] proposes for unsupervised learning and calls this method Extreme Learning Machine Auto Encoder (ELM-AE). In [47], the authors compare ELM and SVM in the use of the same kernel. ELM provided better results than SVM. In [48], the authors introduce the regularized OS-ELM (ReOS-ELM), which uses Tikhonov regularization for optimization, to address the performance issues of OS-ELM when noisy data is encountered. So far, we discuss different approaches to ELM whether from the perspective of several machine learning methods, RBF, and cross-validation. In this thesis, we investigate the RFF method using SLFN in different ways such as applying cross-validation to RBF kernel parameters and optimizing the Fourier features using SLFN.

We construct an SLFN using Fourier features, and we use the SLFN both to learn the Fourier features and classify them in optimized kernel space. There are studies in the literature that evaluates this subject simultaneously with our work. Hence, these studies are superficial, and this thesis offers an investigation in detail. Besides, we show the steering application of this approach [49] in which we discuss the related work for steering in a wireless mesh network in Chapter 4.

3. LEARNING OF FOURIER FEATURES WITH A NEURAL NETWORK

Binary classification to classify data with binary output, e.g., recognizing a visual object as a car or human, has been widely studied in machine learning [50]. There are various methods to classify data represented by $\{(x_t, y_t)\}_{t=1}^N$: $x_t \in R^{1 \times d}$ (d is the dimension of the data) into the classes, i.e., labels, $y_t \in \{1, -1\}$. In general, binary classification divides into two sections; linear classification shown in Fig. 3.1a and non-linear classification shown in Fig. 3.1b. The linear binary classification has robust techniques such as SVM, logistic regression, perceptron [50, 51]. The kernel machines, neural networks (SLFN's), multilayer perceptron, SVM, k-nearest neighbors, naive bayes [52] are several examples of nonlinear classification methods. Our main focus for nonlinear classification methods are kernel machines and SLFN, and we investigate their relations.

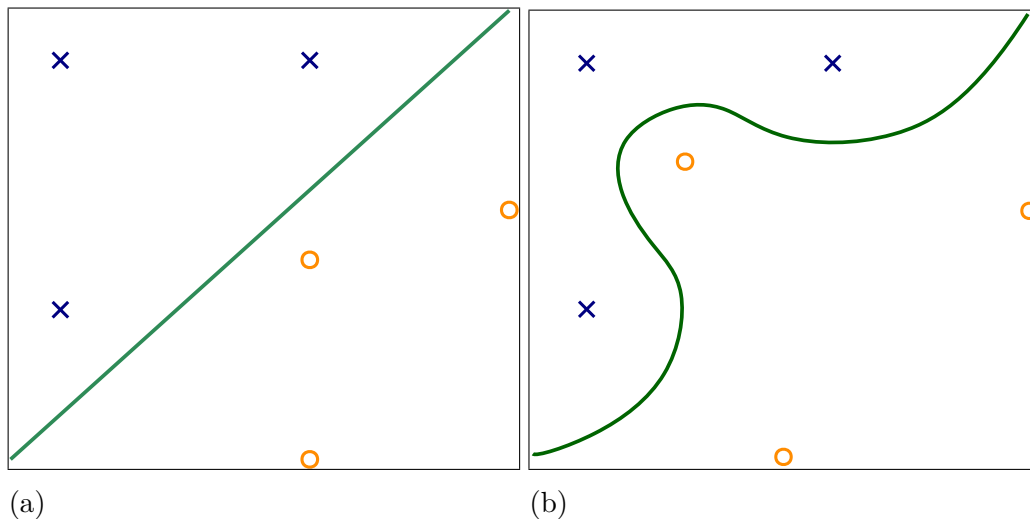


Figure 3.1 An example of linear classification in (a) and nonlinear classification in (b)

In order to achieve nonlinear modeling capability, we use the kernel approach to nonlinear classification in which 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 $z_i^T z_j = k(x_i, x_j)$ and $R^{D \times 1} \ni z = \phi(x)$ is the mapping to the high dimensional space. [53, 54] 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 the high dimensional z space to solve a nonlinear classification problem in the observation x space. This approach is also known as the kernelization of linear techniques or simply "kernel trick", cf. Fig. 3.2 for a visual interpretation. Furthermore, having defined an appropriate kernel function 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) $h(x) = \sum_{i=1}^{N_{sv}} \gamma_i k(x, x_i) + \beta$ 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, having constructed the mapping $\phi(\cdot)$ explicitly appears to be the key to designing techniques that are computationally efficient while benefiting the power of kernels.

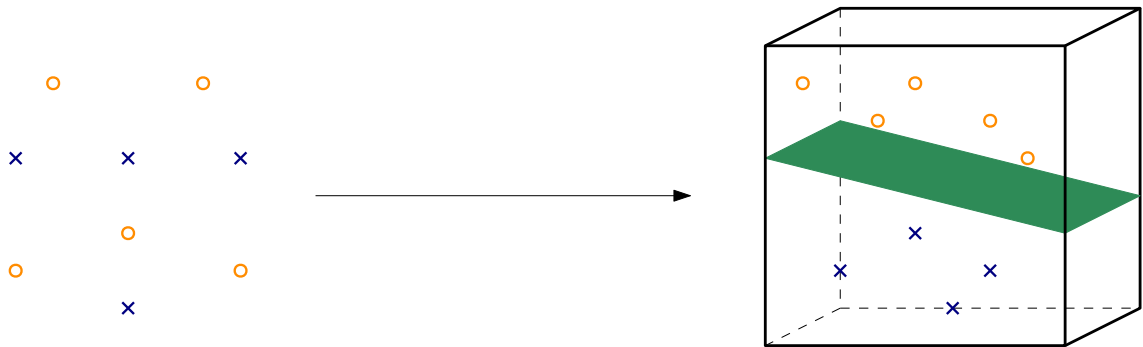


Figure 3.2 Visual interpretation of kernel trick that allows nonlinear classification with linear techniques

Rahimi's suggestion provides a huge advantage in terms of online learning for which the data receive sequentially in time, time-indexed as (x_t, y_t) . At each time instance, an instance of the feature receive, and then the current model updates based on the feedback. Afterward, the received instance discards without being stored in memory which makes the storage complexity significantly small. In this framework of online processing, the computational complexity scales only linearly with the number of processed instances, and thus, scalability achieves with limited storage needs.

On the other hand, Rahimi suggests method does not learn the weights which are chosen randomly in the initial step [3]. Therefore, the efficient model can accomplish by learning those randomly chosen weights which gives better performance. This is where the neural network methods involves in this work. We form the kernel method with deep neural networks and learn the randomly chosen weights using proper learning approach for better classification performance.

To this end, we consider the kernel approach to binary classification and particularly concentrate on random Fourier features [3, 5] for an explicit construction of the kernel space. However, random Fourier features are -in its original proposal [3]- independent of data. For this reason, and based on the recently studied connections between random Fourier features and neural networks in [55, 10], our goal is to investigate various training approaches and algorithms for the learning of Fourier features in the context of neural networks. Afterwards, we present a comprehensive set of experiments with 10 different benchmark datasets, and then demonstrate an application of the learned Fourier features to smart steering (by using the data of [49]) in wireless mesh networks with significantly superior performance compared to [49].

In the following, we explain random kernel expansion in detail.

3.1 Random Kernel Expansion

Random Fourier features (RFF) [3] 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. For any two instances $x_i, x_j \in R^{d \times 1}$ from the feature space, we produce the corresponding set of transformed instances $x_i, x_j \rightarrow z_i, z_j \in R^{2D \times 1}$ such that $k(x_i, x_j)$ can be approximated arbitrarily well by $z_i^T z_j$ as D increases, i.e., $k(x_i, x_j) \simeq z_i^T z_j$.

Remark: This approximation is important because it allows us to exploit computationally efficient online linear learners after the transformation, which is equivalent in principle to training a highly powerful kernel classifier, e.g., SVM with the rbf kernel, in the original space. Note that the idea behind using the kernel is to replace the conventional similarity, i.e., dot products in the original space (thanks to that most linear techniques do only rely on dot products), with an appropriate similarity measure encoded by a kernel (under Mercer’s conditions [2]). This approximation

directly embeds that information into an explicitly constructed known space.

We choose the ‘‘rbf’’ kernel

$$k(x_i, x_j) = \exp(-g\|x_i - x_j\|^2)$$

where g is the bandwidth parameter. In the batch classification analysis part of the presented study, the bandwidth parameter g is determined with 5-fold cross validation, and used in the online classification part as is without a need to update. Also, we emphasize that the method we present can be applied to any shift invariant kernel under Mercer’s condition.

The approximation $k(x_i, x_j) \simeq z_i^T z_j$ is based on random Fourier projections as an application of the Bochner’s theorem (Rahimi, 2008, [3]). Note that the rbf kernel is shift invariant and symmetric, i.e., $k(x_i, x_j) = k(0, x_i - x_j) \triangleq \bar{k}(x_i - x_j) = \bar{k}(-x_i + x_j)$. Then, Bochner’s theorem reads

$$\begin{aligned} \bar{k}(x_i - x_j) &= \int_{R^{d \times 1}} p(w) \exp(jw^T(x_i - x_j)) dw \\ &= \int_{R^{d \times 1}} p(w) \left(\cos(w^T(x_i - x_j)) + j \sin(w^T(x_i - x_j)) \right) dw \quad (\text{due to Euler's identity}) \\ &= \int_{R^{d \times 1}} p(w) \cos(w^T(x_i - x_j)) dw \quad (\text{since } k(x_i, x_j) \text{ is real and symmetric}) \\ &= E_w[\cos(w^T(x_i - x_j))], \\ &= E_w[\cos(w^T x_i) \cos(w^T x_j) + \sin(w^T x_i) \sin(w^T x_j)], \\ &= E_w[r_w(x_i) r_w^T(x_j)], \\ &\simeq z_i^T z_j, \end{aligned}$$

where $r_w(x) = [\cos(w^T x), \sin(w^T x)]$ with

$$(3.1) \quad z_t = \frac{1}{\sqrt{D}} [r_{w_1}(x_t), r_{w_2}(x_t), \dots, r_{w_{D-1}}(x_t), r_{w_D}(x_t)]^T \in R^{2D \times 1},$$

but we use the mapping $Z_w(x) = \sqrt{2} \cos(w^T x + b)$ with b is chosen from uniformly from $[0, 2\pi]$ in this chapter (this mapping also satisfy the condition $E_w[r_w(x_i) r_w^T(x_j)] = k(x_i, x_j)$ [3]), $p(w)$ is a proper probability density, thus, $E_w(\cdot)$ is the expectation with respect to the multivariate Gaussian density $p(w) = N(w; \mu, \Sigma)$ with zero mean $\mu = 0$ and covariance $\Sigma = 2gI$ (I is the identity matrix of the appropriate size),

$$(3.2) \quad z_t = \frac{1}{\sqrt{D}} [Z_{w_1}(x_t), Z_{w_2}(x_t), \dots, Z_{w_{D-1}}(x_t), Z_{w_D}(x_t)]^T \in R^{D \times 1},$$

is the transform with the desired approximation holding due to the law of large

numbers (sample mean approximating the expectation), and $\{w_1, w_2, \dots, w_D\}$ is a set of i.i.d sampled generated from $p(w)$.

Remark: The density $p(w)$ is chosen Gaussian for the reason by Bochner’s theorem that it is the Fourier transform of the rbf kernel. The presented approach works for any shift invariant and symmetric kernel under Mercer’s condition. Hence, for another kernel, one would need to first compute the Fourier transform to specify the corresponding density. We continue with the rbf kernel in the rest.

Remark: Hence, we have obtained, explicitly and in a randomized fashion, the high dimensional space (which is actually infinite dimensional) implied by the rbf kernel. In this high dimensional space, dot products approximate kernel values in the original space so that training a linear classifier in the high dimensional space correspond to nonlinear modeling in the original space as desired. Two advantages for this construct: 1) we can truncate the dimensionality of the transformation to a desired degree that fits to the computational requirements of our application and 2) the rate of convergence is fast; in other words, the quality of the approximation above gets better at an exponential rate in D in accordance with the Hoeffding’s inequality [56]. Thus, D (dimension of the transform) can be chosen relatively small enabling efficiency regarding the computation of the transformation.

Considering our previous example one more time, the decision function of the kernelized SVM $h(x) = \sum_{i=1}^{N_{sv}} \gamma_i k(x, x_i) + \beta$ can now be approximated as

$$(3.3) \quad h(x) \simeq \sum_{i=1}^{N_{sv}} \gamma_i z^T z_i + \beta = z^T \left(\sum_{i=1}^{N_{sv}} \gamma_i z_i \right) + \beta = z^T \alpha + \beta,$$

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 $x \rightarrow z = \phi(x)$. Furthermore, this random mapping with RFF does also allow online processing (one example is presented in [5]) 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 our approach to learning of the Fourier features in order to remove the randomization and design a data driven method.

3.2 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 -as explained in the previous section- 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.

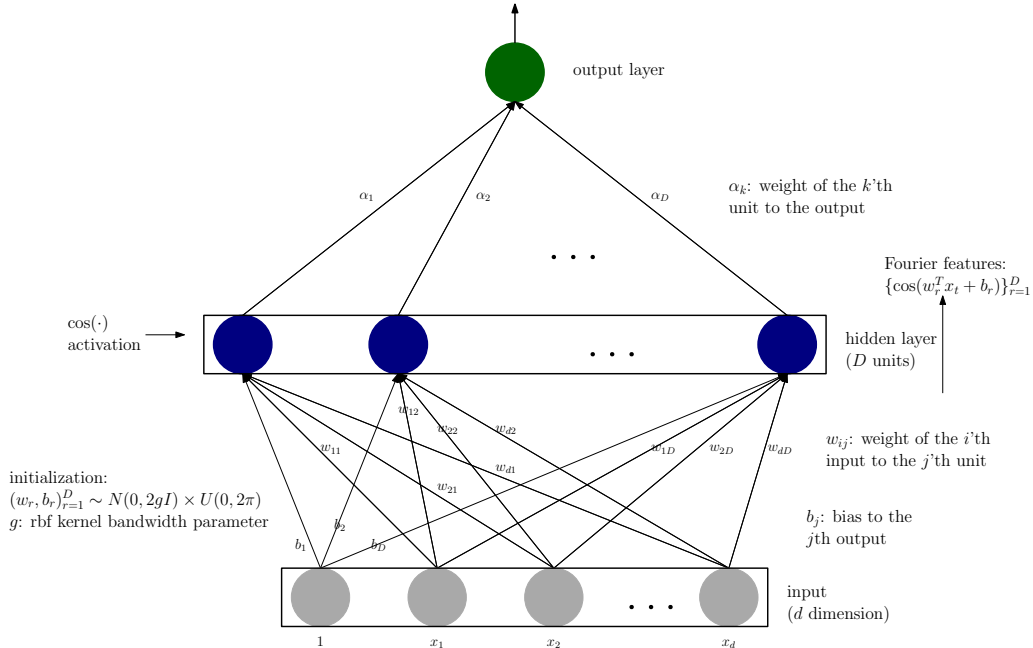


Figure 3.3 Visual representation of our single hidden layer feedforward neural network (SLFN), where d is dimension of the input data instance x and D is the size of the hidden layer implementing the mapping with random Fourier features. The hidden layer activation is sinusoidal producing the Fourier features $\{\cos(w_r^T x_t + b_r)\}_{r=1}^D$. Initially, $(w_r, b_r)_{r=1}^D$ are randomly drawn from the density $N(0, 2gI) \times U(0, 2\pi)$ which is guaranteed to provide -even initially- powerful nonlinear classification as it implements the radial basis function (rbf) kernel. Subsequently, this powerfully initialized SLFN learns Fourier features as its hidden layer activations via the backpropagation based optimization.

Our network in Fig. 3.3 consists of two layers, where the first layer does RFF based random mapping in (3.2). Then the output layer follows with the parameters α and β as in (3.3). 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) $k(x_i, x_j) = \exp(-g\|x_i - x_j\|^2)$ as the kernel, where g is the bandwidth parameter and hence the randomization is Gaussian given by the 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 kernel that

is symmetric and shift invariant.

We emphasize that this observation, i.e., the connection between RFF and neural networks, has been recently made in [55, 10]. However, both studies [55, 10] exploit the mapping in (3.1) to obtain the hidden layer with $2D$ hidden units. In contrast, we exploit the mapping in (3.2) and as a result obtain a more compact network with D hidden units. Another difference is that we opt to learn one magnitude for the output layer and one phase for the hidden layer per each Fourier feature in contrast to learning two magnitudes for the output layer in [55, 10] per each Fourier feature. Since magnitudes are from an unbounded space and phase is from a bounded interval, we consider that our setting is more advantageous in terms of training and stable gradients. This advantage is in addition to the aforementioned benefit of having a more compact network with D units.

We also emphasize that even if the hidden layer of this SLFN (Fig. 3.3) 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 initialization to the subsequent training phase. On the other hand, training the hidden layer optimizes the edge weights $\{(w_r, b_r)\}_{r=1}^D$, which yields the learning of Fourier features as the hidden layer activations $\{\cos(w_r^T x_t + b_r)\}_{r=1}^D$.

In the following, we investigate several training approaches in the introduced context of learning Fourier features with SLFNs, and also provide a baseline for comparisons.

3.3 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, i.e., online kernel learning and SVM with rbf kernel), b) 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 with SGD or minibatch and cross entropy (CE) or mean square error (MSE) losses.

3.3.1 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 principally to the large scale online kernel learning in [5]. In addition, SVM with the rbf kernel or linear SVM in the kernel space expanded by random Fourier features [7, 3] also fall in this category, since a margin based classifier is trained in the kernel space without an attempt to optimize the kernel space.

3.3.2 Fourier Feature Selection (FFS)

This approach -as another baseline for our comparisons- follows an alternative to the neural network based Fourier feature learning, and it is similar in nature to the feature combination with boosting presented in [57]. In this approach, a typically large set of Fourier features are first randomly drawn as previously described, and then useful Fourier features are selected in a greedy manner with the forward selection algorithm. This approach is to compare the compactification power, i.e., to investigate whether the neural network or feature selection performs favorably with the same number of Fourier features.

3.3.3 Two Layer Learning (TL)

This approach is the typical neural network training setting with SGD or minibatch [53]. Both the hidden and output layer are trained.

3.3.4 Batch-Based Two Layer Learning (TL-B)

This training approach processes the data minibatch by minibatch iteratively, in a coordinate descent type optimization framework [58]. 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.

3.3.5 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 [58] with more robust derivatives.

We next present our experiments, where we extensively investigate Fourier features based on these training approaches to binary classification. In particular, we first conduct a performance analysis with 10 different benchmark datasets from various fields.

3.4 Experiments

The benchmark of 10 classification datasets that we use in this part can be found in [1] and summarized in Table 3.1. Each dataset is z-score standard-

ized (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 as well as the cross validation 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 $g \in \{0.01, 0.02, 0.04, 0.08, 0.15, 0.3, 0.5, 1, 2, 4, 8, 10, 20, 30, 40, 50\}$. The optimal values (resulting from cross validation) in each case of the tested algorithm are given in Table 3.2. Minibatch size is 100 and the learning rate is 0.01 in all of the experiments.

Table 3.1 Benchmark details as provided in [1]

Data	Dimension	# of Instances (+,-)	# of epochs for 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 Indians 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 Data Set
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

Based on our overall classification results that are reported in Table 3.3, our observations are as follows: 1) cross entropy (CE) loss yields better results compared to the loss of mean square error, 2) TL approach 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 SGD and minibatch. Also, a joint learning of the 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 approach trained based on the CE loss and the SGD optimization since it is observed to outperform the others.

Table 3.2 Cross validation results: average bandwidth parameter g (upper) and number D of units in the hidden layer (lower) with the corresponding standard deviations in each case

	TL (CE)	SL (CE)	TL-E (CE)	TL-B (CE)	TL (MSE)	SL (MSE)	TL-E (MSE)	TL-B (MSE)
Australian	SGD	SGD	SGD	SGD	SGD	SGD	SGD	SGD
	0.09 ± 0.03	0.11 ± 0.03	0.14 ± 0.06	0.10 ± 0.04	0.12 ± 0.07	0.16 ± 0.09	0.14 ± 0.06	0.14 ± 0.06
	28 ± 30.72	112 ± 75.82	41 ± 40.25	73 ± 50.26	276 ± 164.86	404 ± 420.97	373 ± 414.68	213 ± 184.88
	mini batch	mini batch	mini batch	mini batch	mini batch	mini batch	mini batch	mini batch
	0.08 ± 0.00	0.10 ± 0.03	0.07 ± 0.03	0.09 ± 0.02	0.06 ± 0.03	0.10 ± 0.10	0.09 ± 0.04	0.06 ± 0.03
882 ± 466.89	742 ± 596.78	936 ± 516.32	619 ± 472.31	686 ± 500.22	264 ± 223.59	356 ± 178.58	476 ± 362.68	
Banana	SGD	SGD	SGD	SGD	SGD	SGD	SGD	SGD
	1.00 ± 0.00	1.20 ± 0.42	1.40 ± 0.51	1.10 ± 0.31	1.70 ± 0.48	1.20 ± 0.42	1.70 ± 0.48	1.80 ± 0.42
	27 ± 15.10	78 ± 23.94	46 ± 17.19	40 ± 19.84	150 ± 52.70	180 ± 42.16	142 ± 62.85	170 ± 48.30
	mini batch	mini batch	mini batch	mini batch	mini batch	mini batch	mini batch	mini batch
	1.20 ± 0.42	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00	1.20 ± 0.42	1.00 ± 0.00	1.10 ± 0.31	1.10 ± 0.31
72 ± 51.82	122 ± 56.13	122 ± 56.13	106 ± 54.20	84 ± 66.53	86 ± 47.18	74 ± 54.20	84 ± 67.19	
Breast Cancer	SGD	SGD	SGD	SGD	SGD	SGD	SGD	SGD
	0.11 ± 0.04	0.12 ± 0.03	0.10 ± 0.04	0.11 ± 0.04	0.13 ± 0.09	0.18 ± 0.09	0.15 ± 0.08	0.12 ± 0.10
	175 ± 301.97	208 ± 172.61	208 ± 184.80	166 ± 187.86	285 ± 303.58	376 ± 346.12	156 ± 151.84	222 ± 152.57
	mini batch	mini batch	mini batch	mini batch	mini batch	mini batch	mini batch	mini batch
	0.08 ± 0.03	0.15 ± 0.05	0.10 ± 0.03	0.10 ± 0.03	0.16 ± 0.10	0.14 ± 0.07	0.14 ± 0.10	0.14 ± 0.07
560 ± 333.99	304 ± 266.13	500 ± 294.39	550 ± 134.16	403 ± 436.39	357 ± 385.67	222 ± 313.38	357 ± 385.67	
Diabetes	SGD	SGD	SGD	SGD	SGD	SGD	SGD	SGD
	0.16 ± 0.07	0.17 ± 0.07	0.15 ± 0.08	0.18 ± 0.10	0.17 ± 0.11	0.25 ± 0.07	0.19 ± 0.09	0.17 ± 0.09
	20 ± 22.92	52 ± 28.01	45 ± 47.19	37 ± 48.89	231 ± 251.96	341 ± 327.12	228 ± 212.50	312 ± 275.71
	mini batch	mini batch	mini batch	mini batch	mini batch	mini batch	mini batch	mini batch
	0.10 ± 0.03	0.13 ± 0.06	0.10 ± 0.03	0.10 ± 0.03	0.07 ± 0.04	0.12 ± 0.08	0.11 ± 0.09	0.11 ± 0.04
359 ± 261.19	386 ± 308.36	519 ± 314.09	519 ± 314.09	218 ± 298.83	56 ± 56.11	126 ± 103.40	148 ± 119.20	
Fourclass	SGD	SGD	SGD	SGD	SGD	SGD	SGD	SGD
	1.20 ± 0.42	1.20 ± 0.42	1.70 ± 0.48	1.20 ± 0.42	1.20 ± 0.42	1.20 ± 0.42	1.40 ± 0.51	1.40 ± 0.51
	176 ± 51.46	180 ± 42.16	172 ± 59.02	200 ± 0.00	126 ± 66.70	98 ± 57.69	98 ± 76.15	77 ± 72.56
	mini batch	mini batch	mini batch	mini batch	mini batch	mini batch	mini batch	mini batch
	1.50 ± 0.52	1.10 ± 0.31	1.30 ± 0.48	1.30 ± 0.48	1.00 ± 0.40	0.81 ± 0.31	1.10 ± 0.51	1.10 ± 0.51
166 ± 55.81	190 ± 31.62	190 ± 31.62	190 ± 31.62	74 ± 49.93	88 ± 64.08	66 ± 56.48	66 ± 56.48	
German Numer	SGD	SGD	SGD	SGD	SGD	SGD	SGD	SGD
	0.08 ± 0.02	0.13 ± 0.06	0.10 ± 0.04	0.10 ± 0.03	0.12 ± 0.07	0.13 ± 0.07	0.08 ± 0.05	0.11 ± 0.05
	28 ± 26.56	44 ± 31.08	50 ± 55.55	38 ± 24.29	144 ± 84.66	242 ± 345.96	178 ± 126.08	149 ± 91.07
	mini batch	mini batch	mini batch	mini batch	mini batch	mini batch	mini batch	mini batch
	0.05 ± 0.01	0.06 ± 0.03	0.04 ± 0.02	eee ± eee	0.07 ± 0.04	0.04 ± 0.05	0.05 ± 0.05	0.06 ± 0.06
672 ± 247.87	102 ± 142.57	500 ± 349.22	eee ± eee	720 ± 464.27	436 ± 709.93	288 ± 211.05	404 ± 340.31	
Phishing	SGD	SGD	SGD	SGD	SGD	SGD	SGD	SGD
	0.09 ± 0.03	0.07 ± 0.01	0.08 ± 0.00	0.08 ± 0.00	0.09 ± 0.04	0.11 ± 0.03	0.11 ± 0.03	0.08 ± 0.02
	116 ± 89.52	327 ± 194.45	232 ± 129.02	113 ± 39.42	419 ± 409.90	776 ± 681.66	776 ± 681.66	599 ± 635.31
	mini batch	mini batch	mini batch	mini batch	mini batch	mini batch	mini batch	mini batch
	0.04 ± 0.00	0.08 ± 0.00	0.04 ± 0.00	0.04 ± 0.00	1.05 ± 3.14	0.11 ± 0.04	3.05 ± 6.72	2.05 ± 4.18
2108 ± 748.34	1265 ± 897.66	2652 ± 814.10	2040 ± 555.21	1622 ± 2103.48	1779 ± 2713.28	1027 ± 1060.35	2197 ± 2640.60	
Splice	SGD	SGD	SGD	SGD	SGD	SGD	SGD	SGD
	0.08 ± 0.02	0.08 ± 0.02	0.06 ± 0.01	0.07 ± 0.01	0.08 ± 0.04	0.14 ± 0.02	0.07 ± 0.03	0.09 ± 0.03
	45 ± 15.81	192 ± 61.96	138 ± 75.09	93 ± 60.74	132 ± 78.99	336 ± 210.14	162 ± 69.57	198 ± 189.84
	mini batch	mini batch	mini batch	mini batch	mini batch	mini batch	mini batch	mini batch
	0.04 ± 0.00	0.01 ± 0.00	0.04 ± 0.01	0.04 ± 0.01	0.05 ± 0.03	0.09 ± 0.03	0.06 ± 0.03	0.06 ± 0.03
1320 ± 252.98	246 ± 204.35	1380 ± 289.82	1440 ± 419.52	1014 ± 635.26	1473 ± 1995.61	846 ± 670.79	846 ± 670.79	
Svmguide1	SGD	SGD	SGD	SGD	SGD	SGD	SGD	SGD
	0.85 ± 0.24	0.44 ± 0.09	0.63 ± 0.26	0.60 ± 0.21	0.60 ± 0.21	0.48 ± 0.06	0.50 ± 0.00	0.53 ± 0.17
	40 ± 16.19	112 ± 46.86	78 ± 55.43	73 ± 43.07	208 ± 139.58	256 ± 135.05	228 ± 130.65	147 ± 143.80
	mini batch	mini batch	mini batch	mini batch	mini batch	mini batch	mini batch	mini batch
	0.50 ± 0.00	0.40 ± 0.10	0.40 ± 0.10	0.42 ± 0.10	0.40 ± 0.10	0.29 ± 0.16	0.35 ± 0.14	0.33 ± 0.13
172 ± 125.14	276 ± 133.93	304 ± 126.77	264 ± 121.03	220 ± 133.66	100 ± 78.40	115 ± 124.96	154 ± 147.23	
Svmguide3	SGD	SGD	SGD	SGD	SGD	SGD	SGD	SGD
	0.16 ± 0.08	0.18 ± 0.08	0.18 ± 0.06	0.15 ± 0.05	0.15 ± 0.08	0.19 ± 0.07	0.14 ± 0.06	0.19 ± 0.09
	48 ± 47.23	44 ± 24.40	48 ± 50.90	64 ± 48.12	492 ± 359.14	341 ± 234.64	441 ± 603.54	601 ± 360.23
	mini batch	mini batch	mini batch	mini batch	mini batch	mini batch	mini batch	mini batch
	0.08 ± 0.00	0.06 ± 0.03	0.07 ± 0.03	0.07 ± 0.01	0.16 ± 0.10	0.19 ± 0.07	0.19 ± 0.09	0.18 ± 0.08
609 ± 320.01	277 ± 362.07	668 ± 346.59	563 ± 362.48	551 ± 338.58	546 ± 396.47	450 ± 377.24	471 ± 381.25	

Table 3.3 Benchmark results of TL, SL, TL-E and TL-B algorithms with CE/MSE loss and minibatch/SGD optimizers on ten different data sets

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

Our benchmark results in Table 3.3 are produced by using a similar setting for all algorithms for fairness. After choosing the best performing TL algorithm (with CE and SGD) as discussed above, we now further optimize it (TL algorithm with CE and SGD) standalone in terms of the number of minibatches and learning rate while comparing to SVM with rbf kernel. The resulting accuracy performance is given in Table 3.4, where the kernel bandwidth parameter g is used as the cross validated choice of Table 3.2. 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 3.4 compared to SVM with rbf kernel) the SL algorithm (cf. Table 3.3). Despite this difference, however, 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 kernel space in which they both train a linear classifier, except that SVM with rbf kernel incorporates the strong max margin concept as a regularizer [7]. Therefore, we consider that the use of max margin regularization in SVM with rbf kernel explains this difference between our observations and also explains the exception (the only relatively low performance of the TL algorithm compared to SVM with rbf kernel) in the case of Splice dataset (cf. Table 3.3). Importantly, due to the comparable performance between the TL algorithm and SVM with rbf kernel, we conclude that learning Fourier features can compensate for the lack of max margin regularization. Furthermore, one can expect to outperform SVM with rbf kernel by also using the max margin regularization along with learning Fourier feature, as demonstrated next. As a result, we conclude that learning of Fourier features is largely beneficial by relying on the comparison between the TL and SL algorithms, as they both do not have the max margin regularization whereas SVM with rbf kernel has.

Table 3.4 Comparison of TL algorithm (CE and mini batch) with SVM (rbf kernel) in terms of classification accuracy

Data (epochs, learning rate)	TL algorithm's accuracy (mean / \pm std dev)	RBF SVM's accuracy (mean / \pm std dev)
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

Our last experiment in this section is to compare learning Fourier features with a) the introduced SLFN (by the TL algorithm) and with b) forward feature selection.

The idea of FFS approach is that the candidate D size Fourier features chosen iteratively using linear regression with regularization (regularization parameter as $\lambda = 100$) among the $10D$ randomly initialized Fourier features. At the first iteration, we choose the Fourier features with a size of 10 that give the least mean square error among the $10D$ set. At the next iteration, the next Fourier features with a size of

10 are chosen, together with the previously chosen 10 Fourier features, which gives the least mean square error again. This approach proceeds until the best D Fourier features are chosen.

The Fourier features chose according to the train set of each data set which is %80 of the data set. Then the training set separated into 5 equal sizes of subset and 4 subsets trained using the SL approach with selected Fourier features and test the trained model on the test set. D and g parameters determined by 5 fold cross-validation for the SL approach used for this process.

Table 3.5 Comparison of two layer learning approach, single layer learning approach, single layer learning with chosen Fourier features (FFS) and linear SVM with Fourier features in terms of the mean and standard deviation of accuracy

Data	TL (mb, CE) accuracy (mean \pm std dev)	SL (mb, CE) accuracy (mean \pm std dev)	with chosen Fourier features (mean \pm std dev)	Linear SVM with chosen Fourier features (mean \pm std dev)
Australian	87.07 \pm 1.27	85.15 \pm 2.17	86.30 \pm 3.01	85.23 \pm 2.70
Banana	89.82 \pm 0.66	89.48 \pm 0.65	89.61 \pm 0.93	90.20 \pm 0.70
Breast Cancer	95.83 \pm 1.90	95.16 \pm 1.61	95.33 \pm 2.04	96.41 \pm 1.24
Diabetes	76.40 \pm 1.40	73.40 \pm 3.14	75.33 \pm 3.14	75.86 \pm 2.10
Fourclass	98.41 \pm 1.14	96.23 \pm 1.98	97.94 \pm 1.18	100.00 \pm 0.00
German Numer	74.75 \pm 2.96	68.45 \pm 3.55	70.50 \pm 4.24	74.25 \pm 2.67
Phishing	92.75 \pm 0.61	89.90 \pm 1.19	86.00 \pm 2.49	95.64 \pm 0.62
Splice	83.88 \pm 1.24	50.05 \pm 5.72	51.20 \pm 5.85	81.38 \pm 4.78
Svmguide1	96.55 \pm 0.37	95.78 \pm 0.83	96.05 \pm 0.60	96.71 \pm 0.44
Svmguide3	79.28 \pm 3.10	72.92 \pm 2.70	73.48 \pm 2.28	78.88 \pm 3.54

In Table 3.5, the accuracy results of the SL algorithm with selected Fourier features and linear SVM with selected weights compared with TL and SL algorithm. The results of the SL with selected Fourier features give better accuracy than the SL but slightly worse results than TL algorithm. This is an expected result since we choose the best D Fourier features from $10D$ random Fourier features set, it should exceed the accuracy results of the SL algorithm. Hence, TL algorithm determines the best Fourier features when its training process is over which the SL with chosen Fourier features accuracy is less than TL algorithms accuracy. The results of linear SVM with selected Fourier features give a baseline for the rest of the results.

4. AN APPLICATION OF THE PROPOSED APPROACH: SMART STEERING FOR WIRELESS MESH NETWORKS

4.1 Introduction

Mobile devices have found a widespread use in almost all aspect of our daily lives: home, work, education and entertainment; and consequently, an increasing number of smart phones and diverse applications have triggered a surge in mobile data traffic that is estimated to account 24 for 63 % of all IP traffic [59].

Among various other alternatives, IEEE 802.11 Wi-Fi is the most widely used wireless technology, and with the introduction of new MIMO modes together with dual band operation (2.4 GHz and 5.8 GHz) in IEEE 802.11ac, Wi-Fi link rates have reached Gbps levels [60]. On the other hand, due to the large attenuation through walls and floors, those promised broadband rates cannot be achieved with single access point (AP) Wi-Fi networks in indoor environments. Nevertheless, the throughput and coverage of single AP Wi-Fi networks can be significantly enhanced thanks to the mesh networks, which enable the dynamic organization and configuration of multiple access points (APs) and multi hop routing [61], [62].

A wireless mesh network typically consists of mesh APs and clients and a gateway node, as illustrated in Fig. 4.1. The figure illustrates an example home mesh network, where the gateway AP is connected to the Internet and the clients access the network via multiple APs, which are connected to each other over the mesh links with different cost values that correspond to, for instance, the airtime metric [63].

In all Wi-Fi deployments including Wi-Fi mesh networks, an uneven distribution of wireless clients among APs results in heavily unbalanced networks that suffer from bandwidth or access problems [64]. Also, portability requires a client to seamlessly transition from one AP to another while moving from location to location. Prior

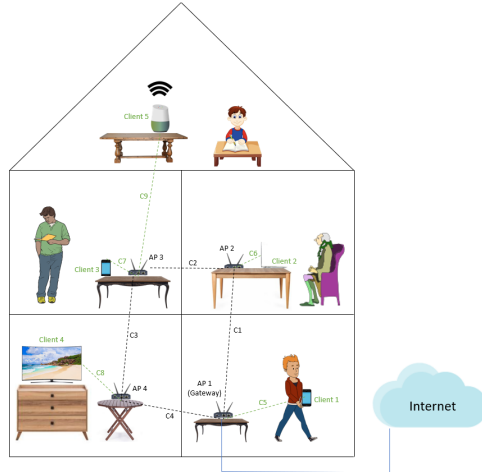


Figure 4.1 An example home mesh network.

to transition, the best target AP needs to be selected, and rather than client-driven operation [65], the transition decision is better managed centrally, as shown in [66]. For multi band mesh networks employing IEEE 802.11ac, the transition decision involves not only moving from one AP to the other, but also involves moving to the appropriate frequency band, which is also referred to as interface. The decision of transitioning a client to a new mesh AP or to a new interface of a mesh AP, is similar to the shortest path routing [67]. That is, the total cost of all possible end-to-end paths from the client to the gateway node is computed by considering the individual costs of connecting to all possible candidate APs and interfaces along the path. Afterwards, the end-to-end path with the smallest cost is selected. The cost metric can involve different link quality metrics, such as Received Signal Strength Indicator (RSSI) or link rate, or it may implement a joint function, as in [67].

In a mesh network, a client can remain persistently connected to an AP even when a better alternative is available. This is known as the sticky client problem, which occurs when the system performs its probing function to observe link metrics, but refuses to roam to the new AP [65]. This might be due to the incorrect or unstable reporting of the link metrics, or directly the vendors of devices that do not perform the probing well. For example, in Fig. 4.1, let the client phone in the second floor be initially connected to AP 3 to access the Internet. Upon being picked up and moved to the room next door, despite computing a smaller cost for connecting over AP 2, if it stays connected to AP 3, then it is said to be stucked to that AP. As a result of the sticky client problem, the enhancements of the mesh network cannot be fully realized, and the persistent connections can starve bandwidth in applications such as video and gaming.

To remedy the sticky client problem, Wi-Fi vendors implement proprietary client

steering solutions, so that a client is directed forcefully to transition to the best possible AP in the network (cf. patents [68], [69]). Certain solutions disconnect the client from the current AP by blacklisting it, while others employ IEEE 802.11v Basic Service Set (BSS) transition management functions [70]. Steering actions do not always successfully produce the intended outcome as they can fail due to –for instance– imperfections in the steering module; hence, the outcome of the steering operation is also not known prior to the attempt.

In this thesis, we present a data driven machine learning approach for analyzing steering modules and identifying when and under which conditions AP-change requests succeed or fail. To this goal, we formulate a binary classification problem based on various network features and train classifiers to learn the critical regions of successful steering actions in the space of network features. Based on this analysis, we propose two smart steering approaches that result in successful transitions. Our contributions and findings can be listed as follows.

- Based on the cloud data collected from a real mesh network employing a client steering module [68], we observe network features, namely current RSSI, current cost, target RSSI, target cost, and we characterize the critical regions of the space of those features, in which the steering actions are more likely to be successful.
- We conduct a batch analysis via Support Vector Machines (SVM) for characterizing the steering data, and based on this analysis, we propose Batch ML for Smart Steering approach.
- Based on the findings of our batch analysis, we develop an online learning approach (online kernel perceptron), namely, Online ML for Smart Steering which is a data-driven, adaptive, real-time algorithm applied to client steering for the first time in the literature.
- The presented online kernel perceptron classifier performs learning sequentially at the cloud from the entire data of multiple mesh networks and operates at APs for steering, both of which are executed in real-time.
- Our results indicate that even by using only two network features, we can classify the steering actions with more than 95% accuracy using our batch ML solution.
- It is shown that the accuracy of our online ML solution is comparable with that of batch ML by a small margin, but the complexity of the online algorithm is significantly smaller by orders of magnitude.

4.1.1 Related Work

A plethora of research has been conducted on handoff and connection management in wireless networks, since the first generation of cellular systems [71]. Topology or graph-based handoff schemes have been studied in [72][73][74][75], where connectivity information is exploited with a focus on minimizing latency. However, client steering is different than the hand off or association problem, as it considers the case when a transition in a Wi-Fi network fails and a client node remains connected to the previous AP. Except for some patented solutions such as, [68], [69] from device vendors, the scientific literature on Wi-Fi client steering, with cloud processing in particular, is rather limited. The authors in [76] propose a bandwidth-oriented handoff steering mechanism, called guided tour of handoff, which assumes that the venue has been surveyed to support indoor localization via fingerprinting based on the received signal strength from reachable APs. Making use of the radio map conveying which APs are available and how strong their signals are at certain locations in the venue, steering is performed to a better AP for the purpose of enhancing throughput.

Machine Learning (ML) based studies in wireless communications and networks have recently received much attention, addressing a wide range of problems from large-scale MIMO systems to device-to-device (D2D) networks, heterogeneous networks to cognitive radio [77]. Since the analysis in our work is based on machine learning via classification; in this section, we review the related ML approaches to the wireless network problems. In [78, 79], K-nearest neighbors algorithms are shown to be beneficial for traffic prediction, for anomaly detection, as well as for modulation classification. Bayesian learning has been applied to channel parameter estimation in [80] and to spectrum sensing in [81]. Support Vector Machines (SVM) is applied for dynamically routing traffic in 5G radio access networks in [82], and for determining the location of wireless nodes in a certain area, by classifying RSSI values, in [83]. In [84], classification is employed in a MIMO wireless network to predict the radio parameters. In [85], for D2D networks, calculation of estimated time of arrivals is improved to obtain the structure of connections, so that a client can determine its time of data transfer based on the network it is connected to. A classification approach can also be used for learning the mobile terminal's specific usage pattern in diverse spatiotemporal and device contexts, as discussed in [86]. Another application is wireless network security, where Denial of Service attacks are detected via classifiers [87]. In such works, APs are classified as authorized and unauthorized based on Round Trip Time values, and unauthorized APs, which spread viruses, hack and steal information from connected clients, are detected and

prevented [88]. In [89], a cognitive radio network is considered and the common control channel for secondary users during a given frame is selected by employing an SVM-based learning technique.

On the other hand, SVM is applied in our work for identification of the critical regions of successful steering actions in a Wi-Fi mesh network. Additionally, based on this analysis, an adaptive, real-time, online smart steering approach is proposed. We emphasize that our targeted application of real time and data driven smart client steering in wireless mesh networks is significantly different than above mentioned studies, and our study is the first to consider client steering from ML perspective under such practical constraints.

4.1.2 Chapter Organization

The rest of the chapter is organized as follows. Section 4.2 presents the problem description. In Section 4.3, the proposed classification based approach for smart steering is introduced, including batch and online steering algorithms. In Section 4.4, our experimental results are presented with the details of considered data and preprocessing, and classification results.

4.2 Problem Description

We study the outcomes of a given steering module as a binary classification problem. Our study is based on the remotely collected data of a Wi-Fi mesh network consisting of multiple clients, APs, and interfaces, where an interface is essentially the frequency band utilized in the connection between the AP and a client. Our collected data (after preprocessing) is a sequence of events (successful or unsuccessful) for each steering action of guidance of a client to another AP or interface. Typically, whenever a client is considered to be better off if it is connected to another access point or interface, then an appropriate steering action, i.e. guidance, is issued. An action does not necessarily result in the intended outcome, generating the labels “successful” (1) or “unsuccessful” (-1), i.e., $Y \in \{-1, 1\}^{N \times 1}$ (N is the number of actions). We associate these labels with “features” that are certain properties of

the network at the time of an action: RSSI and cost of the current as well as the targeted connection of the client, cf. Fig. 4.1 or Fig. 4.2 for an illustration.

RSSI is the received signal strength indicator, whereas cost is an end-to-end metric that provides additional information about the connection quality. Measured between the client and the associated AP or between the root AP with gateway and the other APs, the cost metric per link is often computed, periodically by checking all possible steering connections to evaluate initiating a steering action [68]. We emphasize that the cost feature can be considered as a parameter summarizing the environmental, network- or vendor-specific effects (on steering actions) that are not counted in the remaining features. In our study, the cost per link is calculated as $\frac{50000}{(\lfloor \frac{R}{20} \rfloor + 1)}$, where R is the physical layer link (data) rate, as specified in [68], which is similar to the airtime metric in [63].

Current and target signal strength as well as the costs for the current and targeted connections constitute altogether a four dimensional feature vector for each action, generating the data: $X \in R^{Nx4}$ along with the feedbacks, i.e., labels, $Y \in \{-1, 1\}^{Nx1}$. Our approach is to train a classifier based on this data for obtaining a model $f(x) \in \{-1, 1\}$ in the feature space, which predicts whether an action can succeed or not before it is issued. Note that this classifier suggests a steering outcome on each single point in the feature space, even if it is not tried before by the given steering module. This enables us to explore the whole space and also correct the failing decision points of the existing steering module. Hence, one can consequently decide when exactly to request an action. Namely, one can decide to not follow the existing module; and instead rely on the ones predicted by $f(x)$, which readily defines a novel and data adaptive steering solution that is certainly better (due to training with feedback) compared to rule based nonadaptive ones.

We emphasize that a central agent who requests the steering actions might well be simultaneously observing a large number of mesh networks (for instance, mesh networks of different apartments in a building) each of which might well be serving to multiple clients. For this reason, it is extremely important to devise a model $f(x)$ whose training is computationally scalable to such large scale networks with relatively small space complexity and adaptive to nonstationarity, i.e., possible changes in the network behavior. This realistic processing requirement hinders the use of batch techniques and leads us to consider online algorithms. Therefore, we study the introduced classification problem in the online processing framework in which the data is received sequentially in time (then the data is time-indexed as (x_t, y_t) without a known horizon like N). At each time instance, an instance of steering action is received and then the current model is updated based on the feedback.

Afterwards, the received instance is discarded without being stored so the storage complexity is significantly small. In this framework of online processing, the computational complexity scales only linearly with the number of processed instances, and thus, scalability can be achieved with limited storage needs.

As a result, we sequentially observe $x_t \in R^{4 \times 1}$, and our goal is to learn an online classifier $f_t(x)$ that is sequentially updated at each time with respect to the observed loss $l(f_{t-1}(x_t), y_t)$, i.e., cross entropy loss, induced by the feedback $y_t \in \{1, -1\}$. In addition, we aim to use constant magnitude updates (rather than the conventional update diminishing at a $1/t$ rate) for adaptation to nonstationarity or drifting network parameters. Our proposed solution is a data-driven, adaptive and online, essentially a smart steering module which is a significantly novel contribution to the literature and expected to open-up new directions in managing large mesh networks.

4.3 The Proposed Classification Approach for Smart Steering

We first investigate the learnability of the problem, which is unknown, since the so-called sticky client problem has not been addressed in the literature from the machine learning perspective. One should first answer several ordinary machine learning questions such as 1) which features are most informative, 2) is the problem a linear or nonlinear classification problem, 3) what is the best set of parameters in training a classifier and 4) what is the baseline accuracy for performance evaluation. In order to answer such questions, we first conduct a Support Vector Machines (SVM) [90] based batch analysis (with proper cross-validation and train/test splits for statistical robustness) for the problem, where SVM is chosen as an example for its linear and nonlinear modeling capability and any other algorithm such as Adaboost [2], if desired, can also be used for this purpose. In addition to answering such questions, i.e., the investigation of learnability in the specified machine learning problem, SVM also readily defines a classifier for successful/unsuccessful steering decisions.

In the envisioned system model shown in Fig. 4.2, steering data of a Wi-Fi mesh network consisting of multiple clients, APs, and interfaces is remotely collected at the cloud. SVM analysis considers the entire of the data history collected from all clients as a batch. The resulting classification leads to Batch ML for Smart Steering algorithm (this algorithm is developed/trained at the cloud based on the whole data)

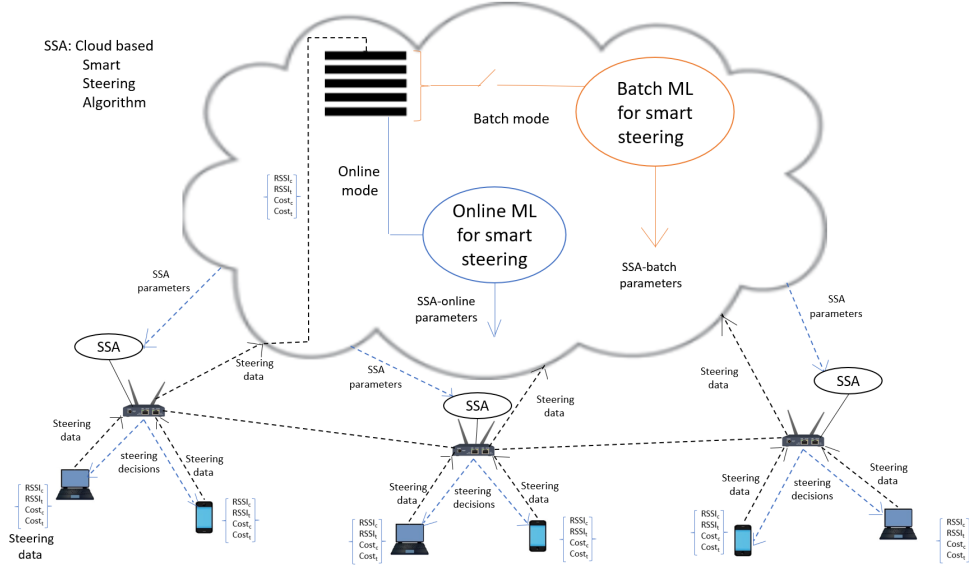


Figure 4.2 Wi-Fi wireless mesh networks with Batch ML and online ML for smart steering.

is executed at each AP (each AP executes the classification model trained at the cloud; hence the communications from the cloud to the APs only require the sharing of the model parameters). Based on the findings of our batch analysis, we develop our online learning [91] approach, named as Online ML for Smart Steering. To this end, we exploit a stochastic gradient descent based minimization of the cross entropy loss [92] (that leads to Rosenblatt’s perceptron [93]) in a compact kernel space [3] for online nonlinear classification, which we call “online kernel perceptron” (but also called as “Fourier online gradient descent” in [5]). The online algorithm, which is again executed at each AP but trained at the cloud based on the whole data, considers only instantaneous values of steering data from all clients (the whole data at the cloud is processed by the online technique sequentially in an online manner without storing, whereas the batch technique stores the whole data and uses it repeatedly). Note that in a mesh network, an AP is able to continuously monitor all of the clients and their features. Therefore, in both cases of the envisioned batch or online scenarios, an AP can decide when exactly to issue a successful steering request (for its own clients), while monitoring the clients, based on the classifier model it received from the cloud. This gives in our work the smart steering.

The details of the proposed batch and online ML smart steering solutions are described next.

4.3.1 Classification Analysis in the Batch Setting

Support Vector Machines (SVM) [2], in its original form [90], is a machine learning algorithm that generates a binary linear classifier based on labeled data $(X, Y) \in (R^{N \times d} \times \{-1, 1\}^{N \times 1})$, where d is the data dimensionality (in our case $d = 4$), N is the number of observations (in our case, number of steering actions), X is the data matrix of features that are also associated with binary labels Y describing the class memberships. SVM essentially learns a separating hyperplane in the feature space to discriminate the two classes. The hyperplane is defined by its normal vector $w \in R^{d \times 1}$ and a bias $b \in R$ such that the resulting classifier is in the form $f(x) = \text{sign}(w^T x + b)$. Then, SVM solves for the hyperplane parameters (w, b) via

$$\begin{aligned} & \min_{w, b, \epsilon} \frac{\|w\|^2}{2} + C \sum_{i=1}^N \epsilon_i \\ & \text{subject to} \\ & y_i(w^T x + b) \geq 1 - \epsilon_i \quad \text{and} \\ & \epsilon_i \geq 0. \end{aligned}$$

This is a convex quadratic programming [2], hence its minimum can be uniquely found. Here, C is a parameter that defines the weight of the training set errors ϵ_i 's (for each instance $x_i \in R^{d \times 1}$) against the (one quarter of the squared) margin $\frac{1}{\|w\|^2}$ which is maximized, ϵ_i 's are referred to as the slack variables that are used to define errors on training set instances based on their distances to the separating plane (an instance on the correct side with decision greater than 1 induces no error, otherwise, the error increases linearly in this cost definition) to solve for data not being linearly separable [7]. The first term of this cost is for the margin maximization and the second term is for the error minimization. The result of this optimization is a separating hyperplane (i.e., equivalently the classifier $f(x)$) that is located between the two classes with maximum margin and least possible errors. The trade-off between these two goals can be controlled by cross validating for C . In this work, we use MATLAB's SVM solver [94].

If one is to generalize SVM to nonlinear class separation, the typical solution is kernelization [2, 7]. Instead of relying on dot products in defining the similarity between two instances x_i and x_j , one typically uses a kernel, e.g., radial basis function (rbf) kernel $k(x, y) = e^{-g\|x-y\|^2}$, to re-define the similarity (g is the bandwidth parameter that leads to more complex (simpler) nonlinear models when relatively higher (smaller) values are set). This strategy implicitly maps the whole data into a high dimensional space where the data is linearly separable. The solution in high dimension then corresponds to nonlinear separation in the original feature space (under Mercer's conditions [2]) as desired.

We point out that SVM is a classification algorithm that works in the batch setting and known to provide decent generalization with especially small scale data. However, it demands multiple accesses to data both in the training and test phases with computational complexities $O(N^3)$ in training and $O(NN_{test})$ in test. This computational complexity of SVM is certainly prohibitive in terms of scalability to large scale data in real-time which is inevitably requested in our targeted application of real-time data driven steering in large mesh networks. Nevertheless, for investigating the classification problem in this thesis, we conduct an SVM analysis first on a small batch to understand whether the problem is linear or nonlinear, which features are most informative, what is the best set of parameters and what is the baseline accuracy provided by SVM. Then, SVM performance is used as a baseline that is competed against in our online setting in favor of computational efficiency while not sacrificing much from the performance. On the other hand, the classifier obtained by SVM can still be used as the Batch ML for Smart Steering for small scale networks in the envisioned system model shown in Fig. 4.2. In this case, the classifier is trained by a central and computationally powerful agent at the cloud based on the whole steering data, whereas the model parameters are shared to the APs and the APs apply the classifier for steering. This limits the communication load between the cloud and the APs while not requiring a heavy computation at the APs since the SVM test complexity is significantly smaller than the SVM training complexity. As a result, an AP can issue successful steering requests for its clients, based on its classifier and based on the real-time-observed client features, for ML based smart steering in this batch scenario.

As it is extensively discussed in our experimental results in Section 4.4, our findings indicate that the problem is not linear but also not severely nonlinear as well. As for the features, we observe that in small scale data, feature pairs perform reasonably well compared to the full dimension. Parameter selection is based on extensive 5-fold cross validation. Finally, the baseline accuracy is observed to be around 95% with nonlinear schemes. The reader is referred for all the details to Section 4.4.

As the problem is identified in Section 4.4 to be nonlinear based on SVM analysis (we use rbf kernel in our experiments for nonlinear modeling), we next discuss an online nonlinear classification algorithm [5]. This online algorithm is comparable with SVM in terms of the accuracy, but it is able to process data significantly faster (compared to SVM) by orders of magnitude. We consider that this is the key to and an early demonstration of smart data driven and online real-time steering that is also scalable to large networks or large number of mesh networks.

4.3.2 Online Classification for Real-time Smart Steering

In this section, we present a method for learning complex, i.e., nonlinear, decision boundaries in the online setting. To this end, we exploit one of the central ideas in machine learning for solving nonlinear classification problems, that is, to lift, i.e., transform, the data to a high dimensional space in which the data is linearly separable and hence linear methods are applicable [2]. In addition, we often do not need to explicitly construct that high dimensional space except the pairwise affinities after the transform. Therefore, it is sufficient to know the inner products that are encoded by a function known as the “kernel”. Nevertheless, kernel techniques (especially in the case of certain kernels such as rbf) in its original form require to compute all of the pairwise kernel evaluations (from test to train) every time an instance is tested, which is usually prohibitively computationally and space-wise complex. Training is even further problematic [3, 57]. This can be addressed by explicitly constructing (though not needed in principle) the high dimensional space through random projections in a compact manner.

We need to guarantee that for online processing, the transformation (i.e. lifting to high dimension) as well as the classifier are both online. We choose perceptron [93] for the online classifier. For the transformation, we use a compact kernel expansion (“compact” means efficient in the sense of the rate of the approximation to the given kernel, c.f. the explanation below) via a random set of Fourier projections [3], which certainly does not disturb online processing.

In the following, we first explain the random kernel expansion, then the application of the perceptron algorithm in the corresponding randomized kernel space and finally

Online ML for Smart Steering in the envisioned system model shown in Fig. 4.2.

Algorithm 1: Online Kernel Perceptron

Input: w_0 is the model parameters for the perceptron in the kernel space, g is the rbf kernel bandwidth parameter (cross-validated), $\{\alpha_i\}_{i=1}^D$ is the set of expansion bases randomly (i.i.d) drawn from $p(\alpha) = N(\alpha; 0, 2gI)$.

Initialize: $w_0 = [0, 0, \dots, 0, 0]^T \in R^{(2D+1) \times 1}$ (last entry is for the bias term).

Initialize: $t=1$.

while *not at the end of the stream* **do**

 Receive the instance x_t .

 Apply the transform:

$$z_t = \frac{1}{\sqrt{D}}[r_{\alpha_1}(x_t), r_{\alpha_2}(x_t), \dots, r_{\alpha_{D-1}}(x_t), r_{\alpha_D}(x_t)]^T \in R^{2D \times 1}.$$

 Make the prediction: $\hat{y}_t = \text{sign}(w_{t-1}^T z_t)$.

 Receive the feedback y_t .

if $\hat{y}_t \neq y_t$ **then**

 | $w_t = w_{t-1} + y_t z_t$

end

$t \leftarrow t + 1$.

end

4.3.2.1 Perceptron in the Randomized Kernel Space: Online Kernel Perceptron

Having the kernel space explicitly and compactly constructed via the transform

$$R^{d \times 1} \ni x_t \rightarrow z_t = \frac{1}{\sqrt{D}}[r_{\alpha_1}(x_t), r_{\alpha_2}(x_t), \dots, r_{\alpha_{D-1}}(x_t), r_{\alpha_D}(x_t)]^T \in R^{2D \times 1}$$

for each instance in a given data stream $\{x_t\}$ (without an end), then a linear classifier in the z -space can solve any nonlinear classification problem provided that the kernel is chosen suitably, e.g., rbf kernel with an appropriate kernel parameter g . For this purpose, we use the perceptron which trains a linear classifier in the online setting through stochastic updates.

A time-indexed online linear classifier $f_t(x) = w_{t-1}^T x + b_{t-1}$ parameterized over (w_{t-1}, b_{t-1}) suffers the loss $l(f_{t-1}(x_t), y_t)$ at time t . If the loss is chosen as the cross entropy loss, i.e.,

$$l(f_{t-1}(x_t), y_t) = -\frac{y_t + 1}{2} \log(\mu_{t-1}) - (1 - \frac{y_t + 1}{2}) \log(1 - \mu_{t-1}),$$

where $\mu_{t-1} = \frac{1}{1+\exp(-\nu f_{t-1}(x_t))}$, then the stochastic gradient update on the model parameters (w_{t-1}, b_{t-1}) yields (w_t, b_t) . This essentially defines the perceptron algorithm (provided that the sigmoid steepness ν is sufficiently large).

We use perceptron after the random kernel expansion based on the transformed stream $\{z_t\}$ for online nonlinear modeling. The resulting “online kernel perceptron” (called as “Fourier online gradient descent” in [5]) is given in Algorithm 1.

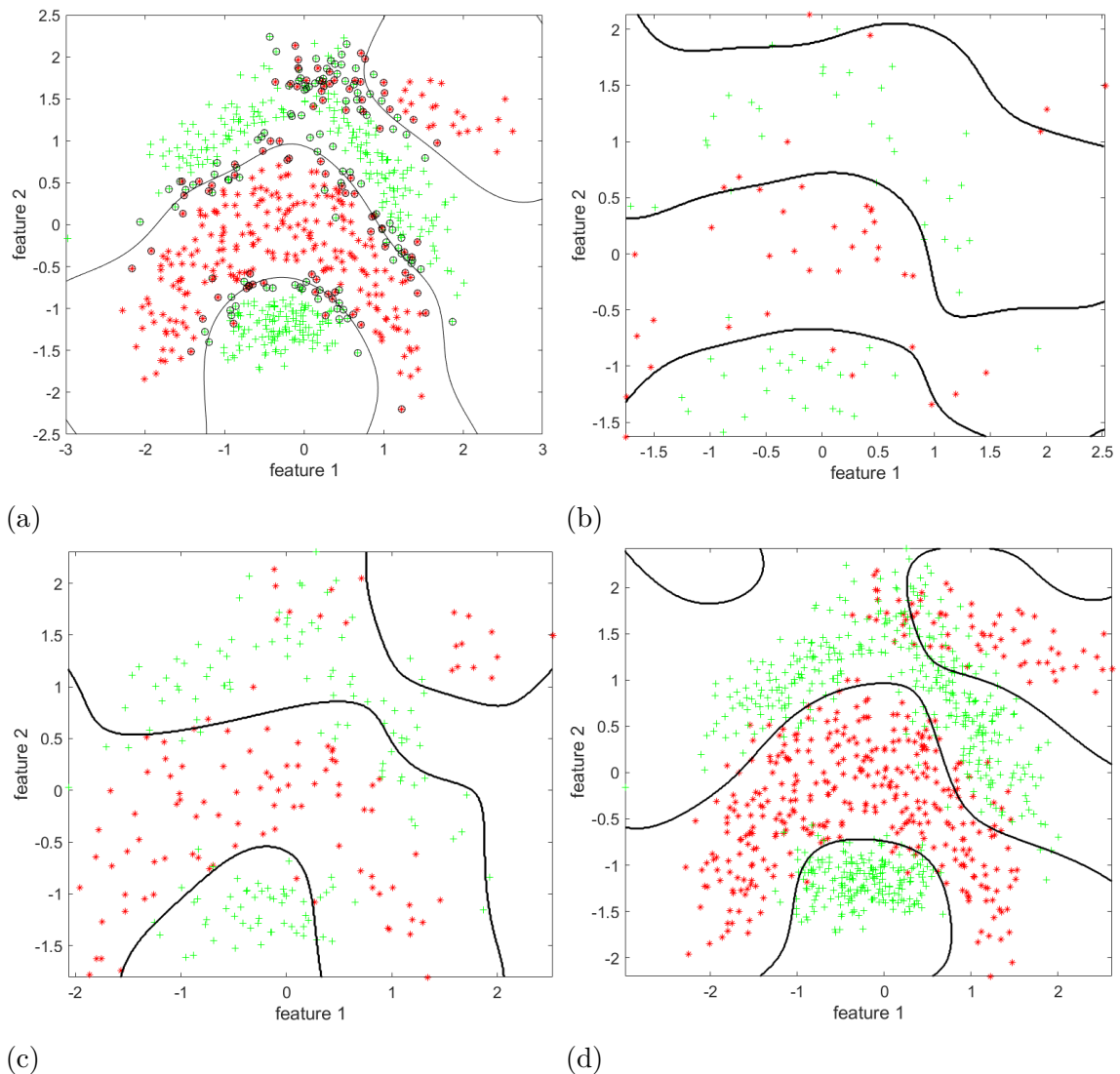


Figure 4.3 The boundary evolution of the banana dataset. In (a), SVM classification with 1000 data instances is presented (error penalty parameter is 8 and kernel bandwidth parameter is 0.7). In (b), (c) and (d), online kernel perceptron classification is presented with 100, 250 and 1000 data instances, respectively.

The presented online kernel perceptron algorithm is computationally highly superior over, for instance, the SVM algorithm as it sequentially processes data with linear complexity whereas the SVM is a batch algorithm. On the other hand, online kernel perceptron is comparable to SVM performance-wise. For a demonstration

on a small synthetic dataset, we present the evolution-in-time of the separation boundary between the two classes in Fig. 4.3. As it is clearly seen when the SVM boundary of the batch setting is compared with the one that the online algorithm is convergent to, the presented online kernel perceptron algorithm is able to learn even complex separations in the sequential setting in a computationally highly superior manner.

We strongly emphasize that the presented online kernel perceptron can be used as Online ML for Smart Steering large scale networks in the envisioned system model shown in Fig. 4.2, in contrast to SVM in the case of Batch ML for Smart Steering for small scale networks. To this end, the online kernel perceptron is sequentially learned in a truly online manner at the cloud based on the whole steering data. The online kernel perceptron is continuously trained at each time at the cloud, and hence, its parameters (the parameters of the resulting linear classifier in the kernel space as well as the parameters of the random kernel transformation) are shared at each time to the APs which execute the received model for steering. This certainly does not require a computationally powerful agent at the cloud or a powerful processor at APs, since both the learning of the online kernel perceptron at the cloud as well as its application at APs are of almost negligible space as well as negligible computational complexity (compared to the batch algorithm). The communication load between the APs and the cloud is still limited (compared to the batch algorithm) since the required number of random Fourier features is relatively small by both theory explained in this section and the practical findings presented in Section 4.4. As a result, successful steering decisions can be made in real-time by APs for all clients in the network by online ML based smart steering as envisioned in Fig. 4.2.

In the following, we present our experimental results.

4.4 Experimental Results

In this section, we present the experimental evaluation of the proposed machine learning based approaches for smart steering in wireless mesh networks. In Section 4.4.1, we introduce the steering data that we use in our experiments. Then, in Section 4.4.2, we present our SVM based batch analysis results along with the comparisons between linear SVM and kernel SVM. Those comparisons conclude that, while yielding the baseline accuracy as given in our accuracy tables of performance

	A	B	C	D	E	F	G	H	I	J	K
1											
2	timestamp	eventtype	eventname	senderMac	targetMac	currentCost	currentRSSI	targetCost	targetRSSI	sequence	
3	1502732071	steering	guided_roam	88:41:fc:12:34:56	88:41:fc:fe:dc:ba	50000	-88	25000	-81	3130229	
4	1502732078	steering	guided_roam_bsstrans_accepted	88:41:fc:12:34:56	88:41:fc:fe:dc:ba	0	-88		-inf		
5	1502732078	steering	guided_roam_bsstrans	88:41:fc:12:34:56	88:41:fc:fe:dc:ba	4	-88		-inf		
6	1502732078	steering	guided_roam	88:41:fc:12:34:56	88:41:fc:fe:dc:ba	50000	-87	25000	-81	3130236	
7	1502732078	steering	guided_11v	88:41:fc:12:34:56	88:41:fc:fe:dc:ba	50000	-87	25000	-81		
8	1502732084	steering	guided_roam_bsstrans	88:41:fc:12:34:56	88:41:fc:fe:dc:ba	4	-87		-inf		
9	1502732084	steering	guided_roam	88:41:fc:12:34:56	88:41:fc:fe:dc:ba	50000	-87	25000	-81	3130241	
10	1502732084	steering	guided_11v	88:41:fc:12:34:56	88:41:fc:fe:dc:ba	50000	-87	25000	-81		
11	1502732085	steering	guide_success_associate	88:41:fc:12:34:56	88:41:fc:fe:dc:ba	3130241	-87		-inf		
12											
13											
14	timestamp	eventtype	eventname	senderMac	targetMac	currentCost	currentRSSI	targetCost	targetRSSI	sequence	
15	1502735393	steering	guided_roam	88:41:fc:12:34:56	88:41:fc:fe:dc:ba	16666	-78	3846	-80	3133004	
16											
17											

Figure 4.4 Logged data from a typical house use for a single AP (a specific window).

results, the classification of steering actions as successful and unsuccessful is nonlinear and it can be solved powerfully with kernel SVM based on certain mesh networks features such as the received signal strength (best set of model parameters have been determined in those comparisons through extensive 5-fold cross-validation). Noting that the SVM classifier is not suitable for real-time processing, we finally introduce the results of our computationally efficient online solutions in Section 4.4.3 for real-time steering.

4.4.1 Steering Data

Our experimental results are based on data recorded in a real Wi-Fi mesh network of typical use of a family house, and include the history of steering actions (event series) of the in-house clients. A sample data window is given in Fig. 4.4.

Note that, each window consists of the event series of a single steering action (out of thousands of actions, i.e., steering decisions). Events in a window are sorted according to their time stamps, forming a time-series. In each row in Fig. 4.4, there is a specific event with the sender and target MAC addresses with additional information about the corresponding access point as well as whether the action is successful. RSSI is a function of the distance between the client and access point, and the cost is calculated by the steering daemon.

From each window, i.e., from each action, we extract a four dimensional feature vector including: (1) RSSI for the current connection (Current RSSI) (2) RSSI for the intended connection for which the action is issued (Target RSSI) (3) cost for the current connection (Current Cost) and (4) cost for the intended connection (Target Cost). We also normalize all feature values to $[0,1]$ by subtracting the minimum and then dividing by the resulting maximum. It is possible to use other features as

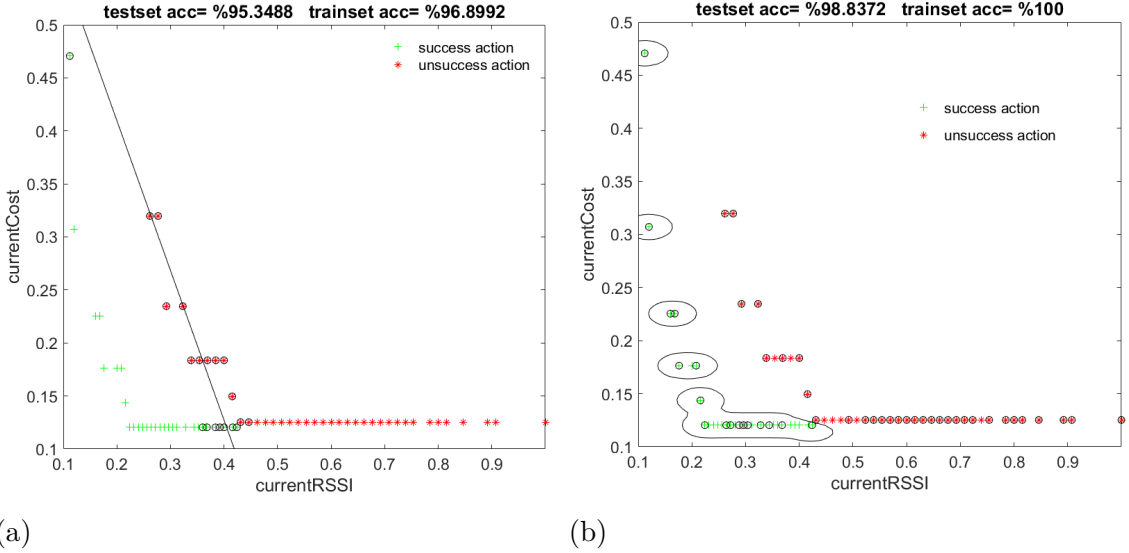


Figure 4.5 (a) Linear SVM classification based on Current Cost and Current RSSI for clients in single AP scenario with transition from 2.4 GHz to 5 GHz and (b) nonlinear SVM classification based on Current Cost and Current RSSI for clients in single AP scenario with transition from 2.4 GHz to 5 GHz.

well, such as the access point/client id or window time duration. However, we opt to use the described feature vector for this study, which is already observed to lead to powerful separation between successful/unsucessful steering actions.

We point out that APs and their two available frequency bands (i.e. interfaces) determine the experimental condition. The mobility of the client can be exchanging the interface between 2.4 GHz and 5 GHz within the same access point or exchanging the access point within the same interface or exchanging both the interface and access point. Hence, a (un)successful action refers to a (un)successful exchange attempt in these cases. As a result, when four dimensional features of steering actions are classified, the steering module can be updated to improve the success rate by using the region of successful transitions.

4.4.2 Results of the Batch Analysis

For classifying successful and unsuccessful steering actions of clients, the cloud data in single and multiple AP scenarios are evaluated based on the client features: Current RSSI, Target RSSI, Current Cost and Target Cost. Linear and nonlinear SVM (with rbf kernel) classifiers have been trained based on different pairs of these features with cross validated parameters (75% of the total labeled data has been used for training, and the remaining part has been used for test. 5-fold cross validation

Table 4.1 Accuracy results of nonlinear SVM in single AP scenario.

	Single AP			
	From 2.4 GHz interface to 5 GHz interface		From 5 GHz interface to 2.4 GHz interface	
	Train set acc.	Test set acc.	Train set acc.	Test set acc.
Current Cost -Target RSSI	100	100	98.40	96.19
Target Cost -Target RSSI	100	98.44	100	99.04
Target RSSI -Current RSSI	98.83	100	99.36	98.09
Current RSSI -Target Cost	95.0325	94.4367	99.68	100
Current Cost -Current RSSI	97.67	96.51	95.54	95.23
Target Cost -Current Cost	92.073	90.906	100	98.09
All four features	100	96.51	100	98.09

within the training set has been used for optimization of the error penalty and kernel bandwidth parameters).

For the single AP scenario, steering of clients between different interfaces of the same AP is considered, steering them from channels of 2.4 GHz band to channels of 5 GHz band, and vice versa. We have applied both linear and nonlinear SVM for classifying successful and unsuccessful actions. As shown in Figures 4.5a and 4.5b, better test set classification is achieved by nonlinear SVM (98.8%), with an improvement of 3% accuracy over linear SVM (nonlinearity in classification can also be observed in Fig. 4.6). Based on our cross validation: in Fig. 4.5a, the error penalty parameter is 1; and in Fig. 4.5b, the error penalty parameter is 32 and the kernel bandwidth parameter is 0.15. For this reason, we opt to continue with nonlinear SVM in the rest of the thesis.

Table 4.1 summarizes the accuracy performance of the nonlinear SVM classifier for a single AP scenario, considering six pairs of client features and two possible transitions from 2.4 (5) GHz interface to 5 (2.4) GHz interface. This table depicts that the achieved test set accuracy is greater than 90% for all cases, with the (Target RSSI, Current RSSI) and (Target Cost, Target RSSI) feature pairs yielding the best classification performance (in the test set) above 98%. Accuracy with all features seems to be less than the one with feature pairs, which is most likely due to the redundancy or feature noise, or the induced data sparsity as a result of increased dimensionality. Consequently, instead of using the 4 features altogether, we choose the pairs yielding the best accuracy and continue our analysis accordingly. (Here, we emphasize that using all of the features is certainly expected to yield a superior or at least comparable performance given sufficient data).

Next, we consider a multiple AP scenario, which involves significantly higher number

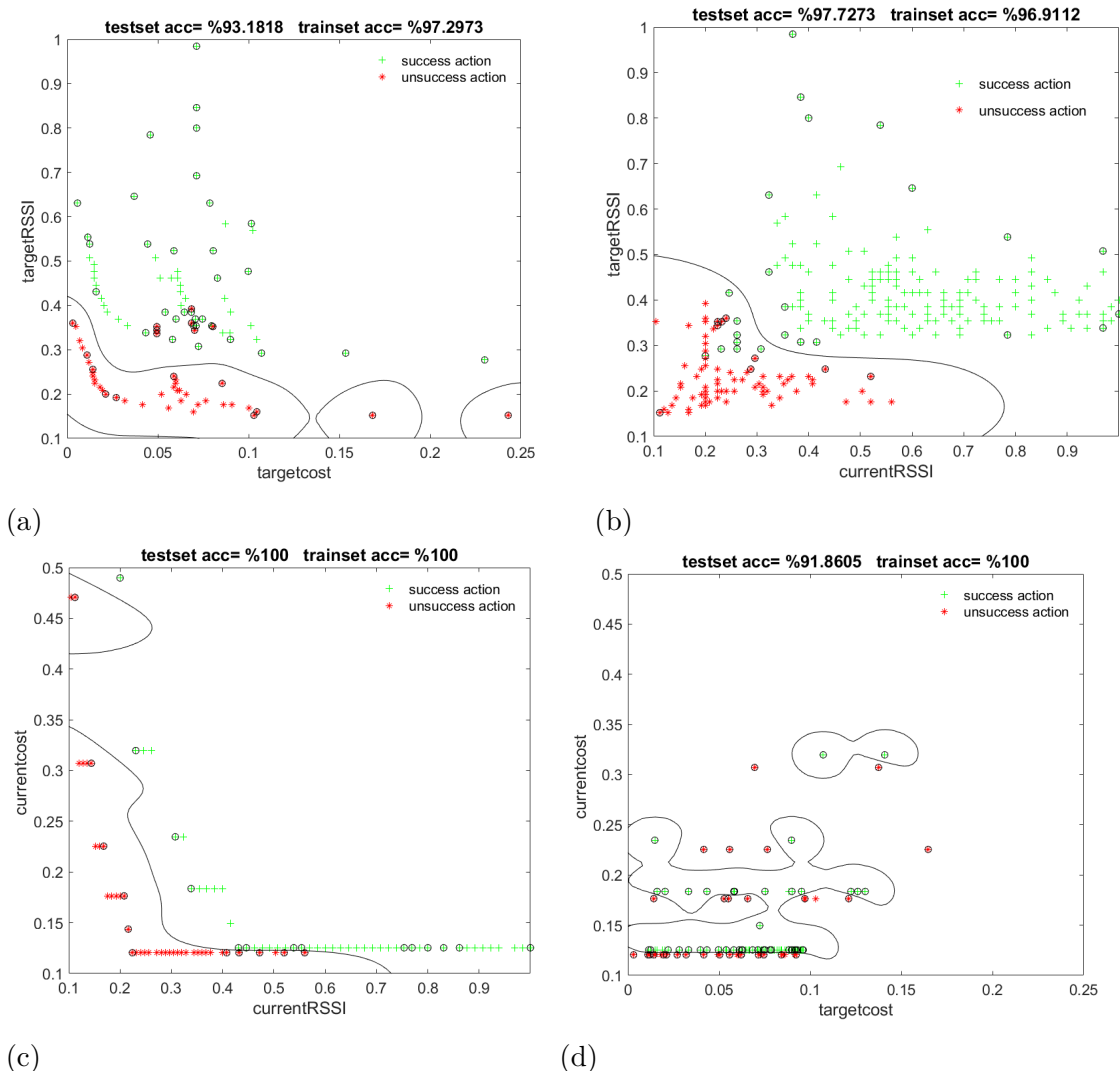


Figure 4.6 Classification results of various feature pairs for steering a client from 2.4 GHz interface of an AP to 5 GHz interface of different AP's.

of steering actions corresponding to transitions between different interfaces within the same AP as well as between different interfaces of different APs. We have filtered the data to include only a single event per steering window by considering only the first steering event per window.

Fig. 4.6 depicts the results of nonlinear SVM classification based on different pairs of client features. It can be inferred from this figure that (Current RSSI, Current Cost) feature pair provides the best (and the most natural) classification with highest accuracy. Unlike this pair, the feature pair of (Current Cost, Target Cost) does not provide a sufficiently good classification when compared to the other pairs. Fig. 4.6 shows that at the same cost level, when the RSSI gets close to 1, i.e., when RSSI increases to -30 dBm (1) from -90 dBm (0), actions become successful. This is expected in such a scenario because clients are in general led to the APs providing a higher signal strength. Based on our cross validation: the error penalty parameter

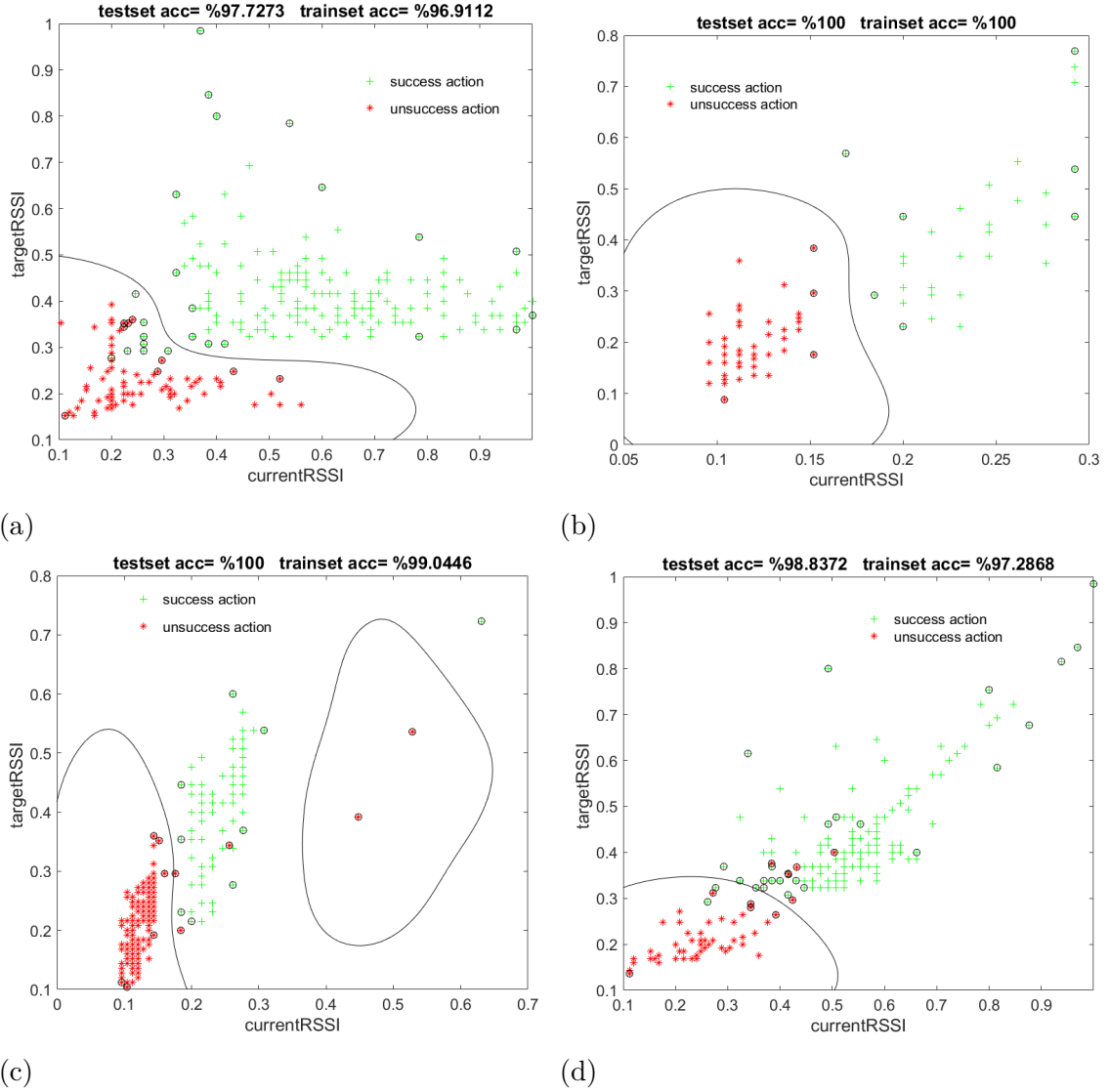


Figure 4.7 Classification results with respect to the (Current RSSI, Target RSSI) pair for steering a client (a) from 2.4 GHz to 5 GHz interface of different AP's (b) from 5 GHz to 2.4 GHz interface of different AP's (c) from 2.4 GHz interface of an AP to 5 GHz interface of the same AP (d) from 5 GHz interface of an AP to 2.4 GHz interface of the same AP.

and the kernel bandwidth parameter are (8,0.5) in Fig. 4.6a, (2,0.8) in Fig. 4.6b, (8,0.4) in Fig. 4.6c and (4,0.2) in Fig. 4.6d.

Experiments in the multi AP scenario also show that different steering transitions can result in different classifier models, since each transition has its own critical region. The main reason causing different critical regions is that the cost computation algorithm varies for different transitions. As an example, in Fig. 4.7, we present classification results based on the (Current RSSI, Target RSSI) pair considering four different steering transitions: (a) from 2.4 GHz interface to 5 GHz interface of the same AP (error penalty parameter is 2 and kernel bandwidth parameter is 0.8),

Table 4.2 Accuracy results of nonlinear SVM in the multi AP scenario.

	Multi AP			
	From 2.4 GHz interface to 5 GHz interface		From 5 GHz interface to 2.4 GHz interface	
	Train set acc.	Test set acc.	Train set acc.	Test set acc.
Current Cost -Target RSSI	94.98	93.18	96.42	92.85
Target Cost -Target RSSI	97.68	95.45	100	100
Target RSSI -Current RSSI	97.29	96.59	100	100
Current RSSI -Target Cost	93.82	82.95	100	96.42
Current Cost -Current RSSI	97.67	96.51	100	100
Target Cost -Current Cost	92.073	90.906	91.66	82.142
All four features	100	95.45	100	96.42

(b) from 5 GHz interface to 2.4 GHz interface of the different AP (error penalty parameter is 2 and kernel bandwidth parameter is 0.8), (c) from 2.4 GHz interface of an AP to 5 GHz interface of another AP (error penalty parameter is 4 and kernel bandwidth parameter is 0.6), (d) from 5 GHz interface of an AP to 2.4 GHz interface of another AP (error penalty parameter is 2 and kernel bandwidth parameter is 0.8). Observing Fig. 4.7, it can be seen that, in plots (a) and (d) Target RSSI and Current RSSI are effective together as a pair for classification. Meanwhile, for transitions plotted in (b) and (c), Current RSSI alone can identify the successful and unsuccessful regions. Overall, despite some outliers, the performance of all nonlinear SVM classifiers provide sufficiently high accuracy, above 96%.

Table 4.2 summarizes the accuracy performance of the nonlinear SVM classifiers for the multi AP scenario, considering six pairs of client features and different transitions for steering (In this table, the results are based on the data of interface transitions from an AP to the same or different AP which result in small mismatches in numbers compared to Fig. 4.6 and Fig. 4.7 where the transitions are confined in the same AP or from an AP to a different one). Again, above 90% accuracy is achieved for all feature pairs (except two cases), and (Target RSSI, Current RSSI) and (Current Cost, Current RSSI) pairs yield highest (above 96%) overall accuracy.

When the results in Table 4.1 and Table 4.2 are compared, one can deduce that the accuracy of SVM classifiers in the single AP scenario is better than the one in the multi AP scenario. This is because, in the multiple AP scenario, APs are placed in various locations, which affect RSSI and cost parameters hence resulting in a more complicated classification problem. The last rows of both tables show the accuracy results for the four dimensional SVM classifier, where SVM creates a hyperplane with four dimensions, providing accuracy over 95%, which is less than the accuracy

of the best pairs. One possible reason is the curse of dimensionality: addition of less effective features creates detrimental effects on the accuracy, when SVM is run with the same amount of data.

The presented results can be exploited to implement a tester for steering algorithms (or profiles) used for directing the clients through APs or interfaces in a Wi-Fi network. The success rate for steering clients (across not only the previously tried actions but across the complete feature space to obtain a better evaluation) can be determined using the critical regions of our classification, and then the best steering profile (among several options) can be selected to achieve the best performance. In this work, for instance, a profile for interface transition from 2.4 GHz to 5 GHz is tested and its success rate for steering has been found to be approximately %73. A new profile with the best success rate can also be created by optimizing the profile parameters.

Note that the SVM classifier separating the steering actions that are successful and unsuccessful can be used to determine the exact time of issuing a steering request during the mobility of a client. Hence, the successful steering of clients can be made possible with the least possible overhead of unsuccessful ones. Given the sufficient number of clients and mobility, the critical regions of the successful actions learned by the classifier can yield the topological structure of the place the network is used in. Also, given two different steering profiles, one can evaluate the both profiles and choose the better one based on the critical regions. Furthermore, such critical regions of successful actions can readily define an optimal smart steering profile, that is machine learning based and hence purely data driven.

As a result of our SVM based batch analysis, we have determined that the classification problem in hand here is nonlinear, the feature pairs (Target RSSI, Current RSSI) and (Current Cost, Current RSSI) provide the best accuracy (at the current level of available data) and the baseline accuracy is around 95%, where the best set of model parameters have been determined via 5-fold cross validation.

4.4.3 Results of Online Classification

As also extensively discussed previously, an SVM classifier (for instance, when employed at a central agent) is not capable of learning from data continuously and hence not capable of real-time monitoring of a possibly cloud based multiple mesh networks serving a total of hundreds of users (cf. Fig. 4.2), and it is even worse

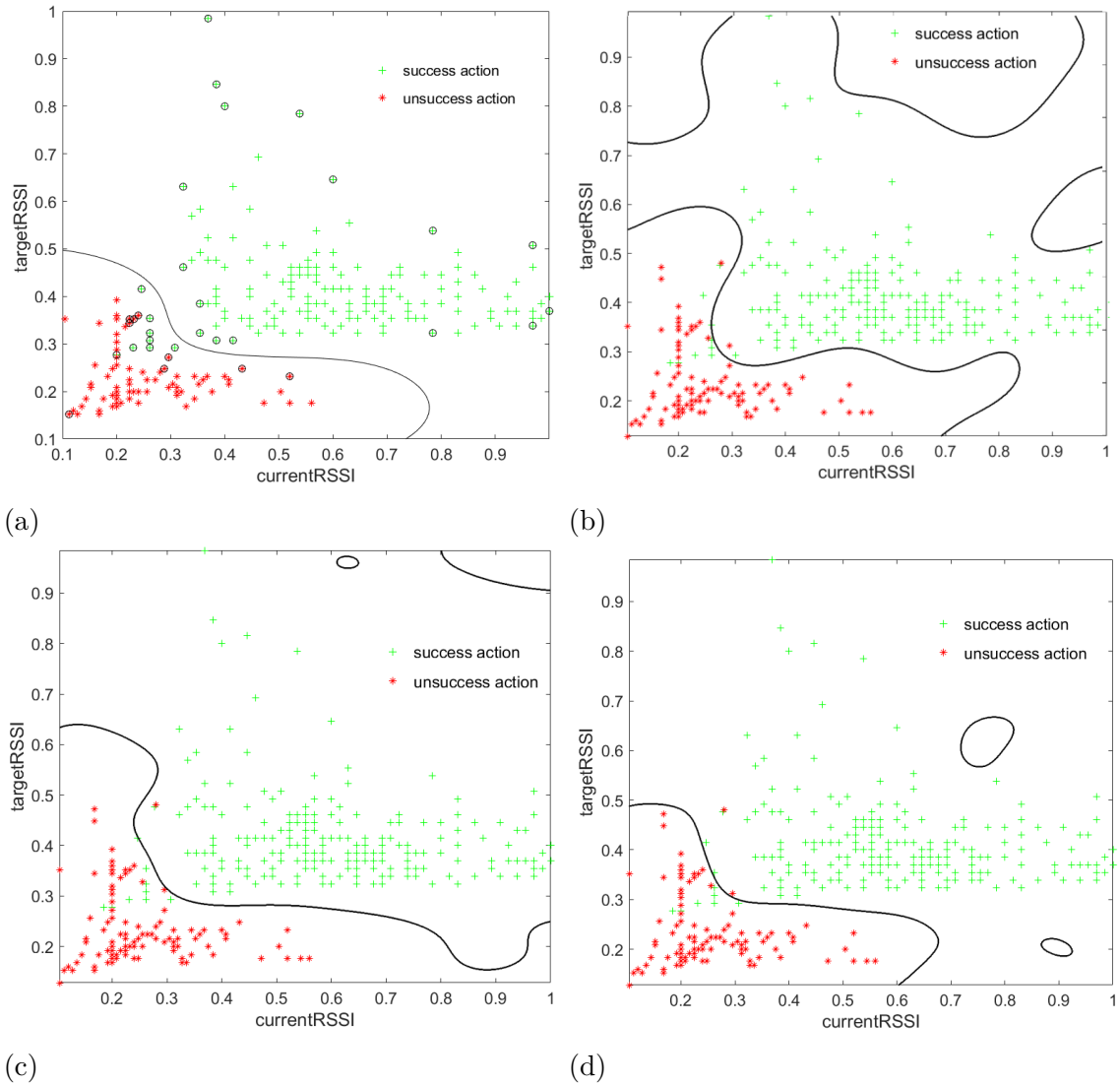


Figure 4.8 The boundary evolution of the dataset of the transition from 2.4 GHz to 5 GHz of different AP's with the feature pair (Target RSSI, Current RSSI). In (a), SVM classification with 1000 data instances is presented (error penalty parameter is 2 and kernel bandwidth parameter is 0.8). In (b), (c) and (d), online kernel perceptron classification is presented with 300, 600 and 1000 data instances (kernel bandwidth parameter is 0.8), respectively.

when the conditions of the networks are highly dynamic and nonstationary requiring online adaptation. The reason is that the computational complexity of SVM scales cubically with the number of training samples which is unbounded in our targeted cloud based scenario in Fig. 4.2, especially when online adaptation is required due to nonstationarity.

To handle this and allow real-time monitoring, we presented the online kernel perceptron algorithm for continuously learning from data even with highly nonlinear class separations. The computational complexity of the presented online algorithm scales only linearly with the number of samples and its space complexity is only con-

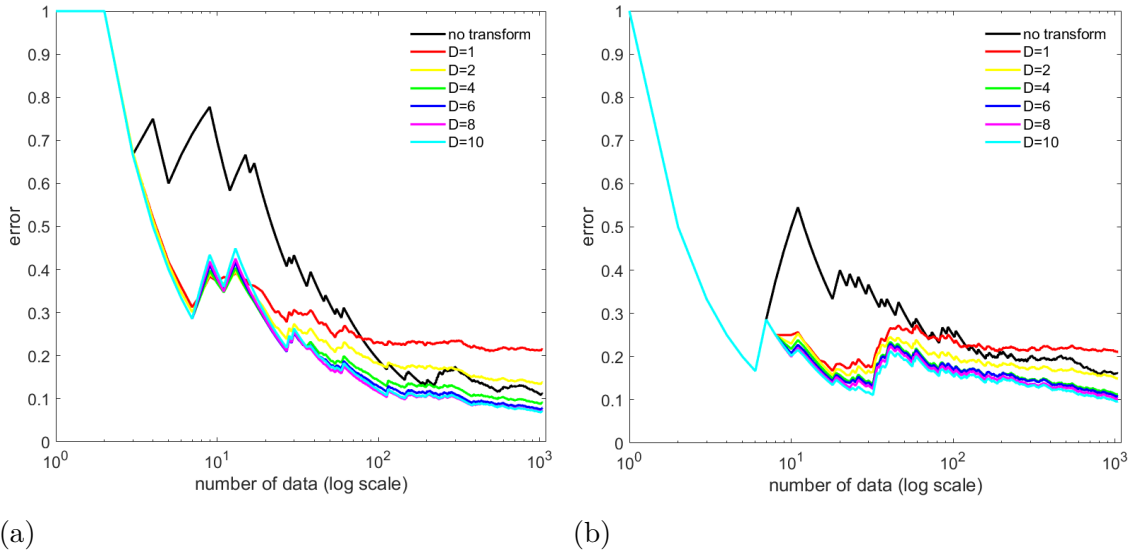


Figure 4.9 Online kernel perceptron classification results based on the feature pairs (Target RSSI, Current RSSI) and (Current Cost, Current RSSI) for clients in multi AP scenario with transition from 2.4 GHz to 5 GHz are given in (a) and (b), respectively, up to $D = 10$.

stant. Moreover, the presented algorithm can adapt to dynamic conditions through stochastic updates.

In the following, our results are used to demonstrate the ability of the presented online kernel perceptron, which drastically cuts down the computational and space complexity, in approximating the baseline provided by SVM analysis. Note that our analysis in this part is confined to the feature pairs (Target RSSI, Current RSSI) and (Current Cost, Current RSSI) as they are observed to provide the best accuracy in our SVM analysis.

We start with a visual presentation in Fig. 4.8, which shows a sequence of class separations during the sequential application of the presented online kernel perceptron in transition from 2.4 GHz to 5 GHz of different AP's with the feature pair (Target RSSI, Current RSSI). We observe that the baseline separation provided by the SVM is very well approximated at around the 1000'th instance with the online kernel perceptron (note that in this case, we have used $D = 50$ for an excellent approximation to the rbf kernel with the bandwidth parameter g matching to the cross validated one of SVM in Fig. 4.8a). This is in accordance with our observation in Fig. 4.3 which shows that the mesh network data is well-behaved from the machine learning perspective.

Next, in Fig. 4.9, we evaluate the accumulated average error rate performance of the presented online kernel perceptron for the feature pairs (Target RSSI, Current RSSI) and (Current Cost, Current RSSI). In particular, we compare the regular

Table 4.3 Error rate results of online kernel perceptron in the multi AP scenario.

	Multi AP			
	From 2.4 GHz interface to 5 GHz interface (for D=50)		From 5 GHz interface to 2.4 GHz interface (for D=50)	
	Same AP	Different AP	Same AP	Different AP
Current Cost -Target RSSI	0.05054	0.1229	0.08091	0.1048
Target Cost -Target RSSI	0.05712	0.1194	0.04249	0.04249
Target RSSI -Current RSSI	0.05737	0.06593	0.04576	0.04956
Current RSSI -Target Cost	0.07806	0.1473	0.05651	0.01919
Current Cost -Current RSSI	0.05645	0.08131	0.05249	0.02004
Target Cost -Current Cost	0.3933	0.4229	0.3362	0.4666

perceptron in the original feature space (this corresponds to the “no transform” case in the figure), which is an online linear classifier, with the online kernel perceptron for various cases $D \in \{1, 2, 4, 6, 8, 10\}$, which is online nonlinear. As D increases, the error performance improves reaching saturation around $D = 6$ while outperforming the regular perceptron (“no transform” case) with $D = 4$ by about 2 – 5% in both cases (note that here, we match the cross validated bandwidth parameter g from our SVM analysis in the rbf kernel used in random Fourier expansion). This, one more time, reinforces that the classification problem we encounter is nonlinear, but not severely as $D = 6$ seems sufficient. Hence, we conclude that the intrinsic dimensionality is still higher than the original one, and only $2 \times D = 12$ number of random Fourier features are sufficient for a powerful classification for our mesh network data.

Finally, in order to evaluate the presented online kernel perceptron algorithm in terms of its capability of approximating the baseline accuracy provided by our SVM based batch analysis, we present our error rates results in Table 4.3. In the case of an excellent approximation to the rbf kernel with $D = 50$, Table 4.3 reports the error rates of the online kernel perceptron based on the data of transition from the interface 2.4 GHz (also from 5 GHz) of an AP to the interface 5 GHz (also to 2.4 GHz) of the same AP or of different APs.

When we compare Table 4.3 with Fig. 4.7, we observe that in the case of, for instance, the feature pair (Current RSSI and Target RSSI), the online kernel perceptron is able to approximate the baseline within a margin of at most 5 – 10% while drastically reducing the complexity. If the results in Table 4.3 are considered together with the ones in Table 4.2 for a general comparison, the online kernel perceptron is observed to maintain its approximation power of the baseline except a few cases.

In Table 4.4, we apply TL algorithm to the steering data set which consists of several pairs of wireless mesh network features. Results are quite good since they

give accuracies over 96% in most of the pairs.

Table 4.4 Accuracy results of TL learning algorithm in the multi access point (AP) scenario

	Multi AP Scenario			
	From 2.4 GHz interface to 5 GHz interface		From 5 GHz interface to 2.4 GHz interface	
	Same AP	Different AP	Same AP	Different AP
Current Cost Target RSSI	96.83 \pm 1.57	90.83 \pm 3.96	93.00 \pm 2.44	93.50 \pm 3.97
Target Cost Target RSSI	98.50 \pm 1.57	95.16 \pm 2.40	97.37 \pm 2.19	99.00 \pm 1.33
Target RSSI Current RSSI	96.00 \pm 1.85	97.00 \pm 1.63	98.87 \pm 0.67	99.00 \pm 1.69
Current RSSI Target Cost	95.00 \pm 1.66	89.33 \pm 4.09	98.00 \pm 2.03	99.50 \pm 0.76
Current Cost Current RSSI	97.00 \pm 4.93	97.16 \pm 2.58	98.87 \pm 1.30	99.16 \pm 2.00
Target Cost Current Cost	74.16 \pm 9.49	74.16 \pm 7.19	94.50 \pm 1.50	98.33 \pm 1.82

4.5 Discussion

Client steering, i.e., requesting changes to the access points (APs) that the clients are connected to, is used in wireless mesh networks to fully realize the performance enhancements promised by the mesh configuration in the network, for instance, by alleviating the sticky client problem. Nevertheless, a change request or a steering action do not always succeed due to several reasons such as the imperfections in the steering module, and hence the corresponding client might suffer from an underperforming connection.

In this chapter, we address and present a data driven machine learning approach for analyzing steering modules and identify when exactly and under which conditions steering actions succeed or fail. This identification can be used to improve the steering module and to develop a novel module that maximizes the overall success probability. To this goal, we formulate a classification problem in both the batch (SVM) and online (kernel perceptron) setting based on various network features and train classifiers to learn the nonlinear critical regions of successful steering actions. In particular, we propose the sequential, continuous and real-time learning of such critical regions at the cloud with the presented online kernel perceptron classifier

based on the whole data of multiple mesh networks serving many clients. Scalability of our approach to such large scale networks is straightforward, since the presented online kernel based algorithm is computationally highly efficient with almost negligible space complexity while being able to learn highly nonlinear models. In the course of the sequential stochastic updates (for improvement in terms of the modeling power of the critical regions of the successful steering actions) to our online algorithm at the cloud, the most recent version of the algorithm at every time can be executed at APs (via the sharing of the relatively few number of model parameters from the cloud to the APs), and this essentially leads to a machine learning based, data adaptive, online and real time smart steering for wireless mesh networks, which we name Online Machine Learning for Smart Steering. In our experiments, we achieve -at least- 95% of classification accuracy in identifying the conditions for successful steering as a result of our batch analysis (SVM). On the other hand, the presented online algorithm (kernel perceptron) is observed to successfully approximate the baseline accuracy provided by the batch analysis within a small margin while drastically cutting down the computational as well as space complexity which yields our real time data adaptive and optimal, i.e., smart, steering strategy.

Acknowledgement: The initial phase of this thesis has been supported by AirTies Wireless Networks.

5. CONCLUSION

We investigate the improvement of optimizing the Fourier features on several binary classification data sets. In addition to this, we introduce a random Fourier feature-based two-layer neural network which transforms the data for kernel trick in the first layer, then apply linear perceptron to classify the data. Four different backpropagation based learning approaches are applied to this network and they are tested by 10 different benchmark datasets with different instance numbers and dimensions. In our experiments, the classification accuracy of optimized Fourier features provides better classification accuracy than the untrained Fourier features. Then we apply our network with the TL algorithm as a learning method to the steering data we previously work on. We observed a remarkable improvement between 1 - 45% on that data set for each Wireless mesh network feature pair so that our proposed algorithm outperformed the online kernel perceptron.

6. Bibliography

- [1] C.-C. Chang, C.-J. Lin, LIBSVM: A library for support vector machines, *ACM Transactions on Intelligent Systems and Technology* 2 (2011) 27:1–27:27, software available at <http://www.csie.ntu.edu.tw/~cjlin/libsvm>.
- [2] R. O. Duda, P. E. Hart, D. G. Stork, *Pattern classification*, John Wiley & Sons, 2012.
- [3] A. Rahimi, B. Recht, Random features for large-scale kernel machines, in: *Advances in neural information processing systems*, 2008, pp. 1177–1184.
- [4] T. D. Nguyen, T. Le, H. Bui, D. Q. Phung, Large-scale online kernel learning with random feature reparameterization., in: *IJCAI*, 2017, pp. 2543–2549.
- [5] J. Lu, S. C. Hoi, J. Wang, P. Zhao, Z.-Y. Liu, Large scale online kernel learning, *The Journal of Machine Learning Research* 17 (1) (2016) 1613–1655.
- [6] B. Bullins, C. Zhang, Y. Zhang, Not-so-random features, *arXiv preprint arXiv:1710.10230*.
- [7] C. J. Burges, A tutorial on support vector machines for pattern recognition, *Data mining and knowledge discovery* 2 (2) (1998) 121–167.
- [8] J. H. McClellan, R. W. Schafer, M. A. Yoder, *Signal processing first*, 2003.
- [9] E. G. Băzăvan, F. Li, C. Sminchisescu, Fourier kernel learning, in: *European Conference on Computer Vision*, Springer, 2012, pp. 459–473.
- [10] J. Xie, F. Liu, K. Wang, X. Huang, Deep kernel learning via random fourier features, *arXiv preprint arXiv:1910.02660*.
- [11] D. J. Sutherland, J. Schneider, On the error of random fourier features, *arXiv preprint arXiv:1506.02785*.
- [12] B. Sriperumbudur, Z. Szabó, Optimal rates for random fourier features, in: *Advances in Neural Information Processing Systems*, 2015, pp. 1144–1152.
- [13] R. Chitta, R. Jin, A. K. Jain, Efficient kernel clustering using random fourier features, in: *2012 IEEE 12th International Conference on Data Mining, IEEE*, 2012, pp. 161–170.
- [14] J. B. Oliva, A. Dubey, A. G. Wilson, B. Póczos, J. Schneider, E. P. Xing, Bayesian nonparametric kernel-learning, in: *Artificial Intelligence and Statistics*, 2016, pp. 1078–1086.

- [15] C.-L. Li, W.-C. Chang, Y. Mroueh, Y. Yang, B. Póczos, Implicit kernel learning, arXiv preprint arXiv:1902.10214.
- [16] T. Yang, Y.-F. Li, M. Mahdavi, R. Jin, Z.-H. Zhou, Nyström method vs random fourier features: A theoretical and empirical comparison, in: *Advances in neural information processing systems*, 2012, pp. 476–484.
- [17] A. Wang, L. Law, X. Miscouridou, M. Mider, S. Ip, Kernel learning via random fourier representations.
- [18] G. Letarte, E. Morvant, P. Germain, Pseudo-bayesian learning with kernel fourier transform as prior, arXiv preprint arXiv:1810.12683.
- [19] C. Wang, J. Yang, L. Xie, J. Yuan, Kervolutional neural networks, in: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 31–40.
- [20] L. Bo, C. Sminchisescu, Efficient match kernel between sets of features for visual recognition, in: *Advances in neural information processing systems*, 2009, pp. 135–143.
- [21] M. Varma, B. R. Babu, More generality in efficient multiple kernel learning, in: *Proceedings of the 26th Annual International Conference on Machine Learning*, 2009, pp. 1065–1072.
- [22] F. Liu, L. Zhou, C. Shen, J. Yin, Multiple kernel learning in the primal for multimodal alzheimer’s disease classification, *IEEE journal of biomedical and health informatics* 18 (3) (2013) 984–990.
- [23] S. Vempati, A. Vedaldi, A. Zisserman, C. Jawahar, Generalized rbf feature maps for efficient detection., in: *BMVC*, 2010, pp. 1–11.
- [24] Z. Hu, M. Lin, C. Zhang, Dependent online kernel learning with constant number of random fourier features, *IEEE transactions on neural networks and learning systems* 26 (10) (2015) 2464–2476.
- [25] P. Bouboulis, S. Chouvardas, S. Theodoridis, Online distributed learning over networks in rkh spaces using random fourier features, *IEEE Transactions on Signal Processing* 66 (7) (2017) 1920–1932.
- [26] F. X. X. Yu, A. T. Suresh, K. M. Choromanski, D. N. Holtmann-Rice, S. Kumar, Orthogonal random features, in: *Advances in Neural Information Processing Systems*, 2016, pp. 1975–1983.
- [27] B. McWilliams, D. Balduzzi, J. M. Buhmann, Correlated random features for fast semi-supervised learning, in: *Advances in Neural Information Processing Systems*, 2013, pp. 440–448.
- [28] F. Li, C. Ionescu, C. Sminchisescu, Random fourier approximations for skewed multiplicative histogram kernels, in: *Joint Pattern Recognition Symposium*, Springer, 2010, pp. 262–271.

- [29] G.-B. Huang, L. Chen, C. K. Siew, et al., Universal approximation using incremental constructive feedforward networks with random hidden nodes, *IEEE Trans. Neural Networks* 17 (4) (2006) 879–892.
- [30] G.-B. Huang, Y.-Q. Chen, H. A. Babri, Classification ability of single hidden layer feedforward neural networks, *IEEE Transactions on Neural Networks* 11 (3) (2000) 799–801.
- [31] Z. Man, K. Lee, D. Wang, Z. Cao, C. Miao, A new robust training algorithm for a class of single-hidden layer feedforward neural networks, *Neurocomputing* 74 (16) (2011) 2491–2501.
- [32] N.-Y. Liang, G.-B. Huang, P. Saratchandran, N. Sundararajan, A fast and accurate online sequential learning algorithm for feedforward networks, *IEEE Transactions on neural networks* 17 (6) (2006) 1411–1423.
- [33] G.-B. Huang, N.-Y. Liang, H.-J. Rong, P. Saratchandran, N. Sundararajan, On-line sequential extreme learning machine., *Computational Intelligence* 2005 (2005) 232–237.
- [34] J. De Villiers, E. Barnard, Backpropagation neural nets with one and two hidden layers, *IEEE transactions on neural networks* 4 (1) (1993) 136–141.
- [35] X. Glorot, Y. Bengio, Understanding the difficulty of training deep feedforward neural networks, in: *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, 2010, pp. 249–256.
- [36] G.-B. Huang, Q.-Y. Zhu, C.-K. Siew, Extreme learning machine: a new learning scheme of feedforward neural networks, in: *2004 IEEE international joint conference on neural networks (IEEE Cat. No. 04CH37541)*, Vol. 2, IEEE, 2004, pp. 985–990.
- [37] Q.-Y. Zhu, A. K. Qin, P. N. Suganthan, G.-B. Huang, Evolutionary extreme learning machine, *Pattern recognition* 38 (10) (2005) 1759–1763.
- [38] W. Deng, Q. Zheng, L. Chen, Regularized extreme learning machine, in: *2009 IEEE symposium on computational intelligence and data mining*, IEEE, 2009, pp. 389–395.
- [39] Y. Miche, A. Sorjamaa, P. Bas, O. Simula, C. Jutten, A. Lendasse, Op-elm: optimally pruned extreme learning machine, *IEEE transactions on neural networks* 21 (1) (2009) 158–162.
- [40] G. Feng, G.-B. Huang, Q. Lin, R. Gay, Error minimized extreme learning machine with growth of hidden nodes and incremental learning, *IEEE Transactions on Neural Networks* 20 (8) (2009) 1352–1357.
- [41] J. Zhao, Z. Wang, D. S. Park, Online sequential extreme learning machine with forgetting mechanism, *Neurocomputing* 87 (2012) 79–89.
- [42] M.-B. Li, G.-B. Huang, P. Saratchandran, N. Sundararajan, Fully complex extreme learning machine, *Neurocomputing* 68 (2005) 306–314.

- [43] G.-B. Huang, C.-K. Siew, Extreme learning machine with randomly assigned rbf kernels, *International Journal of Information Technology* 11 (1) (2005) 16–24.
- [44] H.-J. Rong, Y.-S. Ong, A.-H. Tan, Z. Zhu, A fast pruned-extreme learning machine for classification problem, *Neurocomputing* 72 (1-3) (2008) 359–366.
- [45] N. Liu, H. Wang, Ensemble based extreme learning machine, *IEEE Signal Processing Letters* 17 (8) (2010) 754–757.
- [46] L. L. C. Kasun, H. Zhou, G.-B. Huang, C. M. Vong, Representational learning with extreme learning machine for big data, *IEEE intelligent systems* 28 (6) (2013) 31–34.
- [47] G.-B. Huang, An insight into extreme learning machines: random neurons, random features and kernels, *Cognitive Computation* 6 (3) (2014) 376–390.
- [48] H. T. Huynh, Y. Won, Regularized online sequential learning algorithm for single-hidden layer feedforward neural networks, *Pattern Recognition Letters* 32 (14) (2011) 1930–1935.
- [49] B. Kuskonmaz, H. Ozkan, O. Gurbuz, Machine learning based smart steering for wireless mesh networks, *Ad Hoc Networks* 88 (2019) 98–111.
- [50] N. Cristianini, J. Shawe-Taylor, et al., *An introduction to support vector machines and other kernel-based learning methods*, Cambridge university press, 2000.
- [51] D. W. Hosmer Jr, S. Lemeshow, R. X. Sturdivant, *Applied logistic regression*, Vol. 398, John Wiley & Sons, 2013.
- [52] M. Kuhn, K. Johnson, et al., *Applied predictive modeling*, Vol. 26, Springer, 2013.
- [53] E. Alpaydin, *Introduction to Machine Learning*, 2nd Edition, The MIT Press, 2010.
- [54] K.-R. Muller, S. Mika, G. Ratsch, K. Tsuda, B. Scholkopf, An introduction to kernel-based learning algorithms, *IEEE transactions on neural networks* 12 (2) (2001) 181–201.
- [55] B. Can, H. Ozkan, A neural network approach for online nonlinear neyman-pearson classification, *arXiv preprint arXiv:2006.08001*.
- [56] H. V. Poor, *An introduction to signal detection and estimation*, Springer Science & Business Media, 2013.
- [57] F. Porikli, H. Ozkan, Data driven frequency mapping for computationally scalable object detection, in: *2011 8th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, 2011, pp. 30–35.
- [58] S. J. Wright, Coordinate descent algorithms, *Mathematical Programming* 151 (1) (2015) 3–34.

- [59] C. V. N. Index, Global mobile data traffic forecast update, 2016-2021 white paper.
- [60] E. Perahia, R. Stacey, Next generation wireless LANs: 802.11 n and 802.11 ac, Cambridge university press, 2013.
- [61] I. F. Akyildiz, X. Wang, W. Wang, Wireless mesh networks: a survey, *Computer networks* 47 (4) (2005) 445–487.
- [62] X. Wang, A. O. Lim, Ieee 802.11s wireless mesh networks: Framework and challenges, *Ad Hoc Networks* 6 (6) (2008) 970–984.
- [63] Y. Sarikaya, I. C. Atalay, O. Gurbuz, O. Ercetin, A. Ulusoy, Estimating the available channel capacity of multi-hop ieee 802.11 wireless networks, *Ad Hoc Networks* 10 (6) (2012) 1058–1075.
- [64] N. Aharony, T. Zehavi, Y. Engel, Learning wireless network association control with gaussian process temporal difference methods, in: *Proceedings of OPNETWORK*, 2005.
- [65] K. Schneider, D. Turgut, M. Chatterjee, An experimental study on layer 2 roaming for 802.11 based wlans, in: *World of Wireless, Mobile and Multimedia Networks, 2007. WoWMoM 2007. IEEE International Symposium on a*, IEEE, 2007, pp. 1–6.
- [66] S. Bayhan, A. Zubow, Optimal mapping of stations to access points in enterprise wireless local area networks, in: *Proceedings of the 20th ACM International Conference on Modelling, Analysis and Simulation of Wireless and Mobile Systems*, ACM, 2017, pp. 9–18.
- [67] Y. He, D. Perkins, S. Velega, Design and implementation of class: A cross-layer association scheme for wireless mesh networks, *Ad Hoc Networks* 9 (8) (2011) 1476–1488.
- [68] S. M. Gokturk, A. Akcan, M. I. Taskin, Client steering, uS Patent App. 15/534,772 (Dec. 28 2017).
- [69] S. Laroche, C. T. Hoang, G. Moineau, Band steering, uS Patent 8,655,278 (Feb. 18 2014).
- [70] Ieee standard for information technology– local and metropolitan area networks– specific requirements– part 11: Wireless lan medium access control (mac) and physical layer (phy) specifications amendment 8: Ieee 802.11 wireless network management, *IEEE Std 802.11v-2011 (Amendment to IEEE Std 802.11-2007 as amended by IEEE Std 802.11k-2008, IEEE Std 802.11r-2008, IEEE Std 802.11y-2008, IEEE Std 802.11w-2009, IEEE Std 802.11n-2009, IEEE Std 802.11p-2010, and IEEE Std 802.11z-2010)* (2011) 1–433.
- [71] A. Sgora, D. D. Vergados, Handoff prioritization and decision schemes in wireless cellular networks: a survey, *IEEE Communications Surveys & Tutorials* 11 (4).

- [72] M. Shin, A. Mishra, W. A. Arbaugh, Improving the latency of 802.11 hand-offs using neighbor graphs, in: Proceedings of the 2nd international conference on Mobile systems, applications, and services, ACM, 2004, pp. 70–83.
- [73] S. Pack, Y. Choi, Fast handoff scheme based on mobility prediction in public wireless lan systems, IEE Proceedings-Communications 151 (5) (2004) 489–495.
- [74] A. Mishra, M. Shin, W. Arbaugh, Context caching using neighbor graphs for fast handoffs in a wireless network, in: INFOCOM 2004. Twenty-third Annual Joint Conference of the IEEE Computer and Communications Societies, Vol. 1, IEEE, 2004.
- [75] S. Shin, A. G. Forte, A. S. Rawat, H. Schulzrinne, Reducing mac layer handoff latency in iee 802.11 wireless lans, in: Proceedings of the second international workshop on Mobility management & wireless access protocols, ACM, 2004, pp. 19–26.
- [76] S. Bak, K.-H. Jung, C. Yu, Y.-J. Suh, Guided tour of handoff steering for bandwidth in indoor venues, in: Wireless Communications and Networking Conference (WCNC), 2015 IEEE, IEEE, 2015, pp. 1930–1935.
- [77] J. Wang, C. Jiang, Machine learning paradigms in wireless network association.
- [78] Z. Feng, X. Li, Q. Zhang, W. Li, Proactive radio resource optimization with margin prediction: a data mining approach, IEEE Transactions on Vehicular Technology 66 (10) (2017) 9050—9060.
- [79] N. A. Aslam MW, Zhu Z, Automatic modulation classification using combination of genetic programming and knn, IEEE Transactions on Wireless Communications 11 (8) (2012) 2742—2750.
- [80] C. Wen, S. Jin, K. Wong, J. Chen, P. Ting, Channel estimation for massive mimo using gaussianmixture bayesian learning, IEEE Transactions on Wireless Communications 14 (3) (2015) 1356—1368.
- [81] H. E. Choi KW, Estimation of primary user parameters in cognitive radio systems via hidden markov model, IEEE Transactions on Signal Processing 61 (3) (2013) 782—795.
- [82] R. Tipton, M. Austin, Z. Cui, A. Ohana, Dynamic steering of traffic across radio access networks, uS Patent 8,855,625 (Oct. 7 2014).
- [83] S. Pandey, Location classification accuracy for devices inside and outside of a deployment area, uS Patent 9,485,746 (Nov. 1 2016).
- [84] C. Jiang, H. Zhang, Y. Ren, Z. Han, K. Chen, L. Hanzo, Machine learning paradigms for next-generation wireless networks, IEEE Wireless Communications 24 (2) (2017) 98–105.
- [85] D. D. Testa, M. Danieletto, M. Zorzi, A machine learning-based eta estimator for wi-fi transmissions, IEEE Transactions on Wireless Communications 16 (11) (2017) 7011–7024.

- [86] B. K. Donohoo, C. Ohlsen, S. Pasricha, Y. Xiang, C. Anderson, Context-aware energy enhancements for smart mobile devices, *IEEE Transactions on Mobile Computing* 13 (8) (2014) 1720–1732.
- [87] M. Agarwal, S. Biswas, S. Nandi, Detection of de-authentication dos attacks in wi-fi networks: A machine learning approach, in: *2015 IEEE International Conference on Systems, Man, and Cybernetics*, 2015, pp. 246–251.
- [88] D. Kim, D. Shin, D. Shin, Unauthorized access point detection using machine learning algorithms for information protection, in: *2018 17th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/ 12th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE)*, 2018, pp. 1876–1878.
- [89] K. G. M. Thilina, E. Hossain, D. I. Kim, Dccc-mac: A dynamic common-control-channel-based mac protocol for cellular cognitive radio networks, *IEEE Transactions on Vehicular Technology* 65 (5) (2016) 3597–3613.
- [90] C. Cortes, V. Vapnik, Support-vector networks, *Machine learning* 20 (3) (1995) 273–297.
- [91] N. Cesa-Bianchi, G. Lugosi, *Prediction, learning, and games*, Cambridge university press, 2006.
- [92] C. Bishop, C. M. Bishop, et al., *Neural networks for pattern recognition*, Oxford university press, 1995.
- [93] F. Rosenblatt, The perceptron: a probabilistic model for information storage and organization in the brain., *Psychological review* 65 (6) (1958) 386.
- [94] The Mathworks, Inc., Natick, Massachusetts, *MATLAB version R2015a* (2015).