KERNEL LEARNING AND APPLICATIONS IN WIRELESS LOCALIZATION

BY QIAOJUN WANG

A dissertation submitted to the

Graduate School—New Brunswick

Rutgers, The State University of New Jersey

in partial fulfillment of the requirements

for the degree of

Doctor of Philosophy

Graduate Program in Electrical and Computer Engineering

Written under the direction of

Ivan Marsic

and approved by

_____

_____

_____

_____

New Brunswick, New Jersey

January, 2016

ABSTRACT OF THE DISSERTATION

Kernel Learning and Applications in Wireless Localization

by Qiaojun Wang

Dissertation Director: Ivan Marsic

Recent advances in mobile and pervasive computing have enabled accurate location tracking of users wearing wireless devices indoors, where GPS isn't available. Many indoor WiFi location estimation techniques use received radio signal strength (RSS) values from various access points to track users. In recent years, machine learning techniques have been applied to this application and demonstrated the effectiveness for tracking mobile devices. However, many existing systems suffer from the following problems: (1) lack of labeled data, and (2) non-stationary data distribution. In this thesis, we will describe our kernel learning-based method for solving the these challenges. First, since it can be expensive to collect and label RSS training data in large and complex buildings, we propose a semi-supervised learning approach to learn label-aware base kernels, which are shown to be better aligned to the target comparing to traditional base kernels spanned by the eigenvectors of the kernel matrix (or the graph Laplacian); second, since the data distribution changes constantly as devices change and over different time periods, we propose a transfer learning approach in Reproducing Kernel Hilbert Space(RKHS) to adapt the data distribution changes. Experimental results on real-world benchmark data demonstrate the encouraging performance of our proposed schemes.

Acknowledgements

First and foremost I would like to thank my advisor, Prof. Ivan Marsic, for his guidance, support and kindness throughout my Ph.D study. He provided me the opportunity to conduct research in a field that I truly enjoyed, and he shaped me as an independent researcher. Prof. Marsic is smart, knowledgeable, and inspiring. Being truly fortunate, I got an enormous amount of guidance and support from him, financially, academically, and emotionally.

I am grateful to Dr. Kai Zhang for mentoring me in all phases of my doctoral study. I have learnt a lot, and am still learning, from him. He is such a great role model for me to follow on the road of becoming a researcher. He is a deep thinker, a fantastic writer, and a hardworking scientist. I cherish every mind-blowing discussion with him. I will never forget that he stayed up with me before each deadlines, how he helped me refine the slides of my first oral presentation.

I would like to show my great appreciation to Prof. Waheed Bajwa and Prof. Dario Pompili for all your precious time and insightful, valuable comments, from which I learned a lot and gained the confidence of further improving my dissertation to a solid one. It is my great pleasure to have all of you in my dissertation defense committee.

I wish to thank my parents. They are the persons that always stand in my back giving me their unconditional support. It would be worthy if all my effort is to bring their gratified smiles.

Being with my best friend, soul mate and wife, Yanyan, was the greatest part of these past years. She has been my driving force during stressful times and my endless source of happiness.

Dedication

To my wife Yanyan and my parents.

Table of Contents

Chapter 1

Introduction

1.1 Indoor Localization

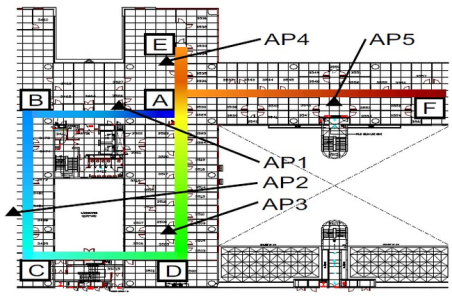
1.1.1 Background

Recent advances in pervasive computing and mobile technology have enabled accurate location and activity tracking of users wearing wireless devices indoors, where GPS isn't available. A practical way to do this is by leveraging the WiFi signals that a mobile client receives from various access points. Many indoor location estimation techniques use received radio signal strength (RSS) values and radio signal propagation models to track users.

As an example, Figure 1.1[57] shows an indoor 802.11 wireless environment of size about 60m by 50m. Five Access Points (AP1,...,AP5) are set up in the environment. A user with an IBM T42 laptop that is equipped with an Intel Pro/2200BG internal wireless card walks through the environment from the location A to F at time tA,...,tF . Then, six signal vectors are collected, each of which is 5-dimensional, as shown in Figure 1.1. Note that the blank cells denote the missing values, which we can fill in a small default value, e.g., 100dBm. The corresponding labels of the signal vectors x_{tA},...,x_{tF} are the 2D coordinates of the locations A,...,F in the building

We can abstract and classify different localization systems by what is carried in the object. Typical models are receiver-oriented model, transmitter-oriented model, transceiver-oriented model and non-intrusive model.[56]

The receiver-oriented model is shown in Figure 1.2 (a), in which the object carries a receiver and receives the messages from different transmitters in the external infrastructure. The object can estimate its distances to these transmitters by measuring the



(a) An indoor wireless environment example.

	AP_1	AP_2	AP_3	AP_4	AP_5
x_{t_A}	-40		-60	-40	-70
x_{t_B}	-50	-60		-80	
x_{t_C}		-40	-70		
x_{t_D}	-80		-40	-70	
x_{t_E}	-40		-70	-40	-80
x_{t_F}	-80			-80	-50

(All values are rounded for illustration)

(b) An example of signal vectors (unit:dBm)

Figure 1.1: An Example WiFi Localization Environment[57]

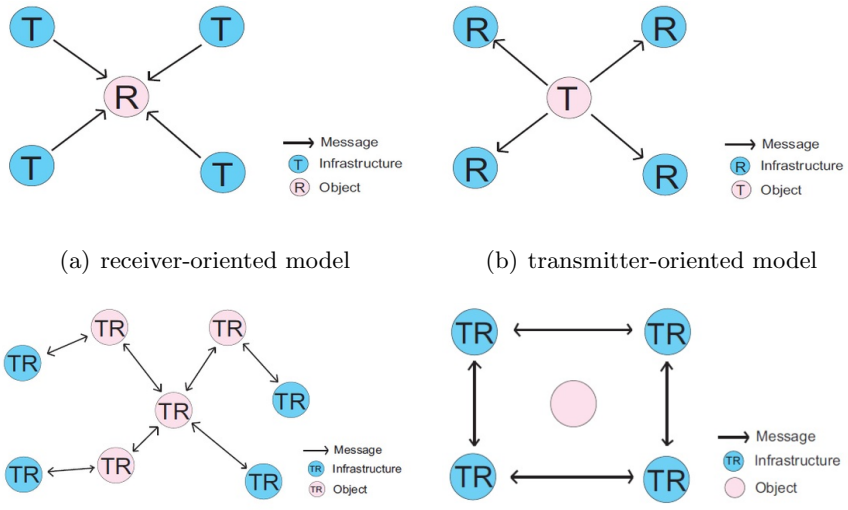
RSS values on received messages. Based on at least three different distance estimations, the object computes its own location. The receiver-oriented model is a natural extension of the GPS mechanism without using timing in-formation. It is mainly used in 802.11-based localization systems, in which the object is usually a person carrying a notebook computer or Personal Digital Assistant (PDA) with 802.11 card and the transmitters are the access points

The transmitter-oriented model, as shown in Figure 1.2 (b), exchanges the roles of transmitters and receivers in the receiver-oriented model. After receiving the messages from the transmitter carried by the object, the receivers in the external infrastructure estimate the distances to the object. By collecting these distance estimations, the external infrastructure computes the location of the object without involving the object in computation. A common scenario for this model is the localization systems that employ Active Radio Frequency Identification (RFID) technique. In RFID-based localization systems, the object carries an active RFID tag periodically broadcasting messages, and the RFID readers are deployed as the receivers. Compared with notebook computer and PDA, a RFID tag has very weak computing capability so that the localization computation must be handled by the external infrastructure

In the transceiver-oriented model shown in Figure 1.2 (c), an object acts both as a transmitter and as a receiver. As the combination of the receiver-oriented and transmitter-oriented model, the object not only estimates distance based on received

messages, but also transmits messages so that other radios can estimate their distances to the object. The common scenario for this model is sensor networks that are commonly dense-deployed without enough infrastructure support. On one side, an object is hard to locate itself within its own neighborhood due to lacking of infra-structure support. On the other side, the object can find many neighbor objects that have their own distance estimation information. It can collaborate with its neighboring objects to solve the localization problem. Furthermore, its neighboring objects may seek the help from their own neighboring transceivers too. In other words, the collaboration could involve of the objects that are multi-hop away. The model reflects the key idea of sensor networks - the collaboration between networked sensor nodes.

In many scenarios such as security surveillance, it is inconvenient or even impossible to require an object to carry a radio. This model, called non-intrusive model, is shown Figure 1.2 (d). Visual tracking falls into this category. For example, multiple cameras may be deployed to cover an interesting area. These camera have to work collaboratively to track an object since each camera may just sense some part of the area.



(a) receiver-oriented model

(b) transmitter-oriented model

(c) transceiver-oriented model

(d) non-intrusive model

Figure 1.2: General Architecture of Tracking Systems [56]

Tracking wireless devices is a difficult task since radio signals usually attenuate in a highly nonlinear and uncertain way in a complex environment where client devices

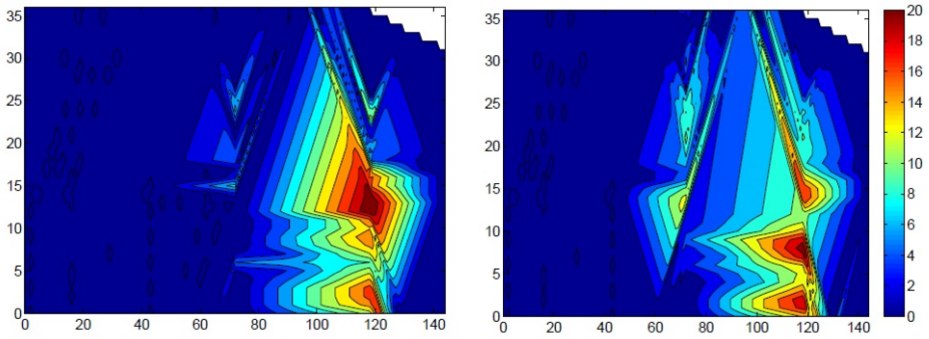
may be moving.

A large class of location estimation systems is based on Learning-based Models, which employ machine learning techniques. Path loss, shadowing and multi-path are caused by a complicated underlying process in a complex environment. When all these factors are mixed together, they show a high-level of nonlinear and noisy patterns. These patterns can be captured when sufficient empirical data are manually collected at different locations or automatically by additional hardware. These methods need less information about physical layout and network configuration. The input is usually implicitly encoded into radio maps at different locations. In such cases, the locations of access points are not needed. Typical pattern descriptions include histogram, mixture of gaussian, kernel matrix, Akima Spline, or simply the mean value of signal strength at different locations. With these algorithms the labels of access points need not be known. Instead, they usually rely on models that are trained with RSS data collected on a mobile device and are labeled with physical locations [53][44][4]. The training data are usually collected offline.

However, on one hand, it is expensive to collect and label RSS training data in a large building because it requires a human to walk with a mobile device, collecting RSS values and recording ground locations. Moreover, RSS data is noisy owing to the indoor environment multipath and shadow fading effects.Therefore, sufficient data shall be collected to power algorithms for approximating the signal to location mapping functions using K-Nearest-Neighbors [4], kernels [4], Bayesian filters [44] and Gaussian processes [58]. A viable approach to compensate for the lack of labeled RSS training data is to design a semi-supervised learning method, where both labeled and unlabeled data are used to boost the learning performance.

On the other hand, RSS data distribution may not always be static over time or across devices[27][93]. First, the data distribution may be a function of time, leaving it difficult to apply a trained model to a new scenario at a different time period. Secondly, the data distribution may be a function of client device, making the model trained for one type of device (say Apple) to be invalid when applied to another device (say Samsung). An example is shown in Figure 1.3[57]. As can be seen, the contours of the

RSS values received from the same access point(AP) at different time periods are very different. Hence, a transfer learning method, which aims to solve the problem when the training data from a source domain and the test data from a target domain follow different distributions or are represented in different feature spaces, is necessary.



(a) WiFi RSS received in T1 from two APs (unit:dBm).

(b) WiFi RSS received in T2 from two APs (unit:dBm).

Figure 1.3: Contours of RSS values over a 2-dimensional environment collected from the same AP but in different time periods. Different colors denote different signal strength values (unit:dBm). Note that the original signal strength values are non-positive (the larger the stronger). Here, we shift them to positive values for visualization.[57]

State of the Art Technologies in Indoor WiFi Localization

The massive deployment of WLANs offers a promising solution for indoor localization. Most of the existing WiFi localization solutions rely on received signal strength (RSS) measurements, and can be divided into two main categories. One family of the localization algorithms based on signal propagation model convert measured signal strength into distance information [50][65]. Propagation model based approaches are widely used for location estimation due to their simplicity and efficiency. These methods usually assume that access points are labeled, e.g., their locations are known. They estimate the distance of the mobile devices relative to some fixed access points based on signal strengths through models that predicts the signal propagation patterns [65]. Researchers have also used Bayesian models to encode the signal propagation

pattern [50] and infer the locations using Monte Carlo methods [82]. A drawback of propagation-model-based methods is that these models may become inaccurate in a complex domain.

On the other hand are those algorithms based on fingerprints[92, 71, 62, 47, 85]. These algorithms mainly consists of two parts, including offline establishment of location fingerprint database and online positioning. At the stage of database establishment, some appointed locations in the building are sampled. A collection of WiFi RSS measurements will be recorded and considered as position fingerprint. At the stage of online positioning, fingerprint information is collected around the position to be localized. Compared with fingerprints in offline database by matching strategy, the position whose fingerprint can attain the best match is chosen as the final estimated position. Although the WiFi fingerprint-based localization technology needs to make fingerprint database at the early stage, it can effectively avoid the influence of building structure. Furthermore, fingerprint-based methods do not require that WiFi access points are known beforehand.

Fingerprinting-based positioning algorithms using pattern recognition techniques are deterministic and probabilistic, K-nearest-neighbor (KNN), artificial neural networks, Bayesian inference, support vector machine (SVM), or their combinations.

Different approaches using WiFi access points are studied from time to time. [60] developed a functional application for a smartphone indoor/outdoor localization system publicly available for download with a name called "Locate Me." It was developed for mobile devices running Android OS and takes advantage of the GPS and WiFi modules to acquire the location of a person. With this system, anybody can find their friends wherever they are. The application sends the current location of the device to the server where it is stored. From that moment on, all friends can access this position and see it on the map. Google Maps Android API is used to represent the users' location, and it has two views available: road view and satellite view. This location system is based on 4 different methods of localization, three for indoor environments and one for outdoors. The fingerprint localization method is used for indoor location.

Indoor localization using WiFi based fingerprinting and trilateration techniques for

location based service is presented by [12]. The paper combined two different WiFi approaches to locate a user in an indoor environment. The first method involves the use of fingerprint matching to compare signal strength data received from nearby access points (AP) by the user, to the reference data stored in the WiFi signal strength database. The second approach uses distance-based trilateration approach using three known AP coordinates detected in the user's device to derive the position. The combination of the two steps enhances the accuracy of the user position in an indoor environment allowing location based services to be deployed more effectively in the indoor environment. An improvement is necessary for finding the correct match for the fingerprinting method with help of incorporating certain database correlation algorithms such as K-nearest-neighbor or probabilistic like a hidden Markov model.

In [2], the authors design a multifloor indoor positioning system based on Bayesian graphical models. In this paper, the author first studied the RSS properties that will affect the overall accuracy of our model like a normal distribution of RSS, using RSS in infer location, and multifloor effect. Markov chain Monte Carlo (MCMC) sampling techniques are used. At the last stage, the author tested their model with four sets of MCMC sampling techniques and compared their results with two well-known location determination systems (RADAR and the Horus). The achieved accuracy is within 3m in a multi-floor environment with a small amount of training points.

By using WiFi, it is possible to define the position of people or assets with good accuracy. In [37], the authors proposed a novel positioning algorithm named predicted K-nearest-neighbor (PKNN) which estimates the current position of amobile user not only by using K found neighbors but also by utilizing its previous positions and speed. In the first stage of the experiment, weighted K-nearest neighbors (WKNN) are used for the position of the tag which is to be estimated. In the second stage, prediction is done for the next probable displacement, based on previous user positions and speeds of Wi-Fi tags. The performance of PKNN for indoor positioning has been evaluated by the experimental test bed. By comparison with KNN, PKNN performs well by 33% or at mean 1.3 meter improvement in error.

A novel, information-theoretic approach is presented in [25] for building a WLAN-based indoor positioning system based on the location fingerprinting system. The proposed technique is based on principal component analysis (PCA) which transforms received signal strength (RSS) into principal components (PCs) such that the information of all access points (APs) is more efficiently utilized. Instead of selecting APs for the positioning which was done by previous researchers, the proposed technique changes the elements with a subset of PCs improvement of accuracy and reduces the online computation. The comparison has been done with AP selection and the Horus system. The proposed approach delivers a significantly improved accuracy. The results show that the mean error is reduced by 33.75 percent and the complexity is decreased by 40 percent.

[22] has been presented, using Dominant AP's RSSI Localization (DARL) algorithms. Using dual log model because of attenuation factor, the parameters are classified into two parts. Firstly, DARL algorithm uses the strongest RSSI from an AP. Secondly, AP trace-back algorithm was suggested as the method for updating the information of unknown AP on the radio map. Optimal filtering system to the proposed algorithm is needed for getting more increased accuracy.

Fingerprinting accuracy performance depends on the number of base stations and the density of calibration points where the fingerprints are taken. Recorded RSSI varies in time, even if there are no changes to the environment. In order to eliminate the deviation of attenuation in the signal, the RSS values are to be averaged over a certain time interval up to several minutes at each fingerprint location. [63] draw on active user participation relying on the contribution of end users by marking their location on a floor plan while recording the fingerprints. The author concludes from long-term measurements over a period of two months that static radio maps cannot be used for room identification even in modest dynamic environments and therefore recommends dynamically adapting algorithms.

[1] proposed an indoor positioning algorithm called WiFi-based indoor (WBI) positioning algorithm. WBI is based on WiFi received signal strength (RSS) technology in conjunction with trilateration techniques.The WBI algorithm estimates the location

using RSS values previously collected from within the area of interest using LSE algorithm, determines whether it falls within the Min- Max bounding box, corrects for non line-of-sight propagation effects on positioning errors using Kalman filtering, and finally updates the location estimation using least square estimation (LSE). The paper analyzed the complexity of the proposed algorithm and compares its performance against existing algorithms. Furthermore, the proposed WBI algorithm achieves an average accuracy of 2.6m.

The authors of [3] proposed a GPS-Like zero configuration indoor positioning system based on received signal strength (RSS) of the popular WiFi network. The proposed system does not require a time-consuming offline radio survey prior knowledge about the area or new hardware unlike current RSS-based indoor systems. Similar to GPS, the proposed system consists of three sections: network segment (WiFi), control segment, and user segment. Between network segment and control segment, RSS observations are exchanged periodically. The control segment uses a novel hybrid propagation modeling (PM) technique using logarithmic decay model augmented by a nonlinear Gaussian process regression (GPR) that models RSS residuals that cannot be modeled by the traditional logarithmic decay models indoors. The proposed system provides 2-3m accuracy in indoor environments.

[95] proposed a wireless indoor localization approach called Locating in fingerprint space (LiFS). In fingerprinting method, radio base requires a process of site survey, in which radio signatures of an interested area are marked with their real recorded locations. Site survey involves intensive costs on manpower and time and is vulnerable to environmental dynamics. The author investigates the sensors integrated in modern mobile phones and user motions to construct the radio map of a floor plan, which is previously obtained only by site survey. On this basis, they design LiFS, an indoor localization system based on off-the-shelf WiFi infrastructure and mobile phones. An experiment was performed in an office building, and the results show that LiFS achieves low human cost, rapid system deployment.

In [81], the authors identified the problem of fingerprint ambiguity and explore the potential to resolve it by leveraging user motion. they proposed a motion-assisted

indoor localization scheme implemented on off-the-shelf mobile phones. It combines user motion patterns with RSS fingerprints to address fingerprint ambiguity. They adopted a crowdsourcing approach to construct a motion database and design a probabilistic algorithm to evaluate location candidates. A prototype is deployed in an office hall covering over 650m2. The localization algorithm limits the mean localization error to less than 1m.

[31] presents a seven-step process involved in building a practical Wi-Fi-based indoor navigation system, which was implemented at the COEX complex in Seoul, Korea, in 2010. More than 200,000 users downloaded the system in its first year of use. Along with Wi-Fi signal-collection sup-port tools, the authors developed a signal and two location filters, and integrated them with the COEX indoor navigation system to make the system more reliable and stable. All of this comprised a seven-step process:(a)Access Point analysis, (b)Design goals Set up, (c)Indoor Map Drawing, (d)Wi-Fi radio Map Construction, (e)System Build-up, (f)System Testing, (g)Service Launching and user Feedback.

Inspired by high densities of smartphones in public spaces, [48] proposed a peer assisted localization approach. The method obtains accurate acoustic ranging estimates among peer phones, then maps their locations jointly against WiFi signature map subjecting to ranging constraints. The authors devised techniques for fast acoustic ranging among multiple phones and built a prototype. It reduced the maximum and 80-percentile errors to as small as 2m and 1m, in time no longer than the original WiFi scanning, with negligible impact on battery lifetime.

[72] explored the WiFi infrastructure to define landmarks (WiFi-Marks) to fuse crowdsourced user trajectories obtained from inertial sensors on users' mobile phones. WiFi-Marks are special pathway locations at which the trend of the received WiFi signal strength changes from increasing to decreasing when moving along the pathway. By embedding these WiFi-Marks in a 2D plane and connecting them with calibrated user trajectories, the method is able to infer pathway maps with high accuracy. The maximum discrepancy between the inferred pathway map and the real one is within 3m and 2.8m for the anchor nodes and path segments, respectively.

[90] designed a self-updating method for the radio map of wireless indoor localization by leveraging mobile devices, which requires no additional hardware or extra user intervention. The authors proposed a trajectory matching algorithm for accurate localization. Their approach globally optimizes the residual errors of an entire trajectory. The authors investigated the static behaviors of mobile devices and exploited their potentials for radio map updating. They prototyped in real environments. When the localization service has run for a long term, AcMu gains accuracy improvement of more than 2 times, compared to using the static original one.

More indoor localization techniques can be found in the recent survey [94]. Recently, WiFi RSS measurements are used together with data from smartphone gyroscope and accelerometer, magnetic field measurements from smartphone magnetometer, and a floor plan of the building for hybrid indoor localization in 2015 Microsoft Indoor Localization Competition [1]. Two winner teams from SPIRIT Navigation and Fraunhofer Portugal Research Center achieved localization error distance around 2 meters.

Several WiFi localization solutions are currently commercialized. Examples of these solutions are Skyhook[2] and Navizon[3]. These two solutions are based on collecting information on WiFi access points and cellular Base Stations locations all over the world and maintaining them in databases. A client location is computed by collecting a raw data and sending it to a location server which returns a location estimate. Google and Apple are both maintaining such databases for providing location-based services on their platforms.

Indoor WiFi Localization Use Case Scenarios

Due to the revolution of localization technologies, location based services have recently attracted significant attention in the spatial aspect of one's life. A large catalog of use cases are proposed based on WiFi localization. We briefly summarize the most popular ones in this section.

---

[1] http://research.microsoft.com/en-us/events/indoorloccompetition2015/

[2] http://www.skyhookwireless.com/

[3] http://www.navizon.com/

- Tracking: It can be used for people tracking: children, patients with dementia, prisoners with arrest ankle bracelets, employers to track their workers, as well for animal tracking.

- Navigation: It allows locating the exact geographical position of a mobile device and get direction and/or navigate user to required location.

- Marketing: (a) Location Triggered Advertisement It uses user location to provide messaging or application alerts based on user preferences and opt-ins. Once opted-in, messages are delivered whenever a consumer enters inside the targeting area. (b) Location Based Social Media It can provide for business opportunity to create an interactive experience in-store?n experience that will convert browsers to buyers and from one-time customers to loyal ones. (c) Local Search Advertising It is advertising for listings of local points of interest (e.g. merchant retailers, restaurants) depending on the location of a mobile device. (d) Proximity Marketing It refers to localized wireless distribution of advertising content associated with a particular place.

- Emergency: One of the fundamental application is utilizing the ability to locate an individual calling to emergency response agency who is either unaware of his/her exact location or is not able to reveal it because of an emergency situation. Based on this spatial information emergency response agency (e.g. ambulance, police, firefighters) can provide help in a quick and efficient way.

- Information Services: It refers mostly to the digital distribution of information based on device location, time specificity and user behavior.

- Sports: It allows user to automatically collect his/her workout data, such as location, distance, speed, duration, or burned calories and store them on the server.

- Billing: Location based billing refers to ability to dynamically charge users of a particular service depending on their location when using or accessing the service.

- Geotagging: Geotagging is defined as adding geospatial metadata to digital media such as photographs, videos, messages, blogs, web pages and GeoRSS.

1.1.2 Semi-supervised Learning

Semi-supervised learning (SSL) is a useful learning paradigm that makes use of unlabeled samples to boost the learning performance with only limited supervision. SSL is halfway between supervised and unsupervised learning. In addition to unlabeled data, the algorithm is provided with some supervision information ?but not necessarily for all examples. Often, this information standard setting will be the targets associated with some of the examples. In this case, the data set X can be divided into two parts: the points $X_l := (x_1, ..., x_l)$, for which labels $Y_l := (y_1, ..., y_l)$ are provided, and the points $X_u := (x_{l+1}, ..., x_{l+u})$, the labels of which are not known. In principle classifiers can have a more accurate prediction by taking into account the unlabeled points. However, there is an important prerequisite: that the distribution of examples, which the unlabeled data will help elucidate, be relevant for the classification problem. The knowledge on $p(x)$ that one gains through the unlabeled data has to carry information that is useful in the inference of $p(y|x)$. If this is not the case, semi-supervised learning will not yield an improvement over supervised learning. It might even happen that using the unlabeled data degrades the prediction accuracy by misguiding the inference.

One common assumption for SSL is so called semi-supervised smoothness assumption. The assumption is that the label function is smoother in high-density regions than in low-density regions.This assumption implies that if two points are linked by a path of high density (e.g., if they belong to the same cluster), then their outputs are likely to be close. If, on the other hand, they are separated by a low-density region, then their outputs need not be close.

A special case of the above semi-supervised smoothness assumption is cluster assumption. It states that if points are in the same cluster, they are likely to be of the same class. This assumption may be considered reasonable on the basis of the sheer existence of classes: if there is a densely populated continuum of objects, it may seem unlikely that they were ever distinguished into different classes. Cluster assumption

does not imply that each class forms a single, compact cluster: it only means that, usually, we do not observe objects of two distinct classes in the same cluster. Cluster assumption can be formulated in an equivalent way, called low density separation. The decision boundary should lie in a low-density region.

A different but related assumption is the manifold assumption: The (high-dimensional) data lie (roughly) on a low-dimensional manifold. It is helpful to tackle the so-called curse of dimensionality problem. If the data happen to lie on a low-dimensional manifold, then the learning algorithm can essentially operate in a space of corresponding dimension, thus avoiding the curse of dimensionality. If we view the manifold as an approximation of the high-density regions, then it becomes clear that in this case, the semi-supervised smoothness assumption reduces to the standard smoothness assumption of supervised learning, applied on the manifold.

Among various directions that have been pursued by researchers, for example, graph based algorithms [9][39], low-density separation [14], transductive SVM [15][51][36], SDP programming [46], ensemble method [45], high order [98], semi-supervised kernel design turns to be a promising one because it allows the abundant theories and algorithms in kernel methods to be adopted directly in solving SSL problems. In particular, a large family of algorithms for semi-supervised kernel relies on spectral transformation, where the eigenvectors of the kernel matrix (or the graph Laplacian) are used together with the rectified eigenvalues to build the new kernel.

Lots of empirical successes have been observed with the family of semi-supervised kernels based on spectral transforms. However, there are still some concerns with them. First, building a kernel solely based on rectifying the kernel eigen-spectrum may be restrictive in terms of acquiring desired kernel. Note that eigenvectors of the empirical kernel matrix (or graph Laplacian) are computed in an unsupervised manner, entirely irrespective of the class labels. They can be inaccurate due to various practical factors such as noise, kernel types or parameters, or class separability. Therefore, these eigenvectors may not reveal useful structures for classification, and the base kernels they span can have low alignment with the target, while the alignment of the mixed kernel depends crucially on the alignment of the individual base kernels. Second, the

optimization procedure involved can be quite expensive. For example, computing the eigenvalue decomposition of the Laplacian already takes $O(n^3)$ time and $O(n^2)$ memory. The time complexity of QCQP ($O(n^4)$) [100] and SDP ($O(n^{4.5})$) [42] is also quite demanding.

To solve these problems, we propose a new way for designing base kernels used in semi-supervised kernel learning. Besides using the eigenvectors from the original kernel matrix or graph Laplacian, we also compute a new set of more "accurate" eigenvectors that are expected to be better aligned to the target. Our key observation is that the kernel eigenvectors and class labels have some intrinsic connections. In particular, the ideal kernel eigenvectors are deemed equivalent as the class labels. Inspired by this, we compute a set of desired kernel eigenvectors by extrapolating the ideal kernel eigenfunction. Such extrapolation builds upon important proximity structures encoded in the input patterns. More importantly, it directly incorporates class labels in the computation. Therefore, the label-aware eigenvectors are empirically more aligned to the target compared with the unsupervised kernel eigenvectors. This directly leads to a set of base kernels with higher quality, and the overall alignment of the mixed kernel will also be improved with better generalization performance. In addition, we use low-rank approximation to compute useful eigenvectors from the original kernel matrix, therefore our approach is computationally very efficient and only requires linear time and space complexities.

It is worthwhile to note that the main contribution is to explore new ways of constructing base kernels, instead of how to combine the base kernels, in semi-supervised kernel learning. The latter has been studied extensively in the literature as has been discussed, and therefore will not be the focus of this work. Of course, in order to evaluate the usefulness of the newly proposed base kernels against traditional base kernels, we will still resort to existing methods of kernel combination so as to finally obtain a mixed kernel and its testing accuracy.

1.1.3 Transfer Learning

In standard supervised learning scenario, it is commonly assumed that the training and the test data are drawn from the same underlying distribution, such that a classifier learned on the former generalizes well to the latter. However, in many practical situations this assumption is violated. For example, the collection of the training and test data can happen under quite different situations in many applications, including bioinformatics, sensor network, and spam filtering; on the other hand, in some cases people might want to apply the knowledge learned from one domain to a different but related domain, if the training procedure is expensive or labels in the new domain are not easy to obtain. In these cases, traditional learning framework is no longer suited, and how to handle the discrepancy of the data distribution in different domains becomes a crucial problem.

Research on transfer learning has attracted more and more attention since 1990s. Transfer learning aims to extract the knowledge from one or more source tasks and applies the knowledge to a target task. In transfer learning, A domain D consists of two components: a feature space $\mathcal{X}$ and a marginal probability distribution $P(X)$, where $X = \{x_1, ..., x_n\} \in \mathcal{X}$. For example, if our learning task is document classification, and each term is taken as a binary feature, then $\mathcal{X}$ is the space of all term vectors, x_i is the ith term vector corresponding to some documents, and X is a particular learning sample. In general, if two domains are different, then they may have different feature spaces or different marginal probability distributions. Given a specific domain, $D = \{\mathcal{X}, P(X)\}$, a task T consists of two components: a label space $\mathcal{Y}$ and an objective predictive function $f(\cdot)$ (denoted by $T = \{Y, f(\cdot)\}$, which is not observed but can be learned from the training data, which consist of pairs x_i, y_i, where $x_i \in X$ and $y_i \in Y$. The function $f(\cdot)$ can be used to predict the corresponding label, $f(x)$, of a new instance x. From a probabilistic viewpoint, $f(x)$ can be written as $P(y|x)$. In our document classification example, $\mathcal{Y}$ is the set of all labels, which is True, False for a binary classification task, and y_i is "True" or "False".

Given specific domains D_S and D_T, when the learning tasks T_S and T_T are different, then either (1) the label spaces between the domains are different, i.e. $\mathcal{Y}_S \neq \mathcal{Y}_T$, or (2) the conditional probability distributions between the domains are different; i.e. $P(Y_S|X_S) \neq P(Y_T|X_T)$, where $Y_{S_i} \in \mathcal{Y}_S$ and $Y_{T_i} \in \mathcal{Y}_T$. In our document classification example, case (1) corresponds to the situation where source domain has binary document classes, whereas the target domain has ten classes to classify the documents to. Case (2) corresponds to the situation where the documents classes are defined subjectively, as such tagging. Different users may define different different tags for a same document, resulting in $P(Y|X)$ changes across different users.

In addition, when there exists some relationship, explicit or implicit, between the two domains or tasks, we say that the source and target domains or tasks are related. For example, the task classifying documents into the categories book, desktop may be related the task classifying documents into the categoriesbook, laptop. This because from a semantic point of view, the terms "laptop" and "desktop" are close to each other. As a result, the learning tasks may be related to each other. Note that it is hard to define the term "relationship" mathematically. Thus, in most transfer learning methods assume that the source and target domains or tasks are related

In this work, we consider the situation when the training data and test data are from different distributions, *i.e.* $P_{tr}(\mathbf{x}) \neq P_{te}(\mathbf{x})$, but are supposed to share some identical or similar conditional distribution $P_{tr}(y|\mathbf{x}) = P_{te}(y|\mathbf{x})$. There have been a number of attempts to solve this problem. Early works include [83, 11], which treated different domains as tasks and applied the multi-task learning. Daumé III and Marcu [35] investigated how to train a general model with data from both a source domain and a target domain for domain adaptation in natural language processing tasks. Recently, several research work [74, 96, 34, 80] has converged along the direction of estimating a point-wise re-weighting on the training data to minimize the generalization error in testing. For example, Huang *et al.* [34] applied the *kernel mean matching* (KMM) to account for the distribution difference, such that the means of the training and test points in a reproducing kernel Hilbert space (RKHS) are close. Sugiyama *et al.* [80, 79]

proposed a framework to estimates the importance ratio which is simultaneously equipped with model selection. Their idea is to find an importance estimate $\hat{w}(\mathbf{x})$ such that the Kullback-Leibler divergence from the true test input density $P_{te}(\mathbf{x})$ to its estimate $\hat{P}_{te}(\mathbf{x}) = \hat{w}(\mathbf{x})P_{tr}(\mathbf{x})$ is minimized. In [59], instead of learning point-wise reweighting coefficients, the authors proposed to learn the so called transfer components by minimizing the Maximum Mean Discrepancy (MMD) criterion defined in the RKHS, which measures the distance between distributions of two samples. The transfer components are in the form of pre-parameterized empirical kernel maps and can handle out-of-samples conveniently.

We note that most of the current methods studies how to make training and testing data have the same distribution in the input space [74, 96, 80, 79]. In contract, few attempts has been made specifically to cater to kernel methods, where being considered should be the data distributions in the reproducing kernel Hilbert space (RKHS). In [34] and [59], although the objective function considered is the difference between the sample mean in the feature space, it is used as an indicator of the distance between two distributions in the input space. Therefore, minimizing such objective is ultimately used to control the difference of distributions in the input space. While in general one may want to consider data distribution in the input space, the behavior of kernel methods are determined in a more complex mechanism due to the interplay between the kernel and the data distribution. In particular, the kernel methods work by applying a linear algorithm in the kernel-induced feature space, where the algorithm performance depends directly on the data distribution in this space. Therefore, we believe that making the training and testing data have similar distributions in the feature space will be a more direct way in tackling the covariate shift problem for kernel-based learning algorithms.

To achieve this goal, we propose to rectify data distribution directly in the Hilbert space by enforcing the closeness on the kernel matrices from the training and testing domains. One big technical difficulty here is that kernel matrices are data-dependent, and how to evaluate similarity between kernel matrices defined from different samples remains unclear. To bridge this gap, we introduce the concept of *surrogate kernels* based

on the Mercer's theorem, the fundamental theorem underlying the reproducing kernel Hilbert space (RKHS). It provides convenient interface for different Gram matrices to communicate and compare with each other. By using the surrogate kernel, we can apply an explicit (linear) transform on the Gram matrix of the training data, forcing it to properly "approach" that of the test data, such that the kernel machine learned on the training data generalize well to test domain. The linear transformation applied on the kernel matrix enforces a flexible, nonlinear mapping in the input space. Our approach acts directly on kernel matrix, the basic building block of kernel machines, and demonstrates satisfactory performance on solving the problem for kernel-based methods.

1.2 Thesis Outline

Chapter 2 briefly reviews kernel-based learning methods and Support Vector Machine(SVM) for regression. Kernel methods emerged in the evolution of machine learning algorithms during mid-1990s, and enabled researchers to analyse nonlinear relations with the efficiency that had previously been reserved for linear algorithms. Kernel-based learning first appeared in the form of SVM for classification. Soon, SVM for regression tasks were developed and obtained excellent performances. Therefor, in our experiments we choose SVM for predicting the location of mobile client in WiFi envrionment.

Chapter 3 presents a semi-supervised learning approach to reduce calibration-effort for tracking a mobile node. Many previous approaches to the location-estimation problem assume the availability of sufficient calibrated data. However, to obtain such data requires great effort. In this chapter, we propose a semi-supervised learning technique for RSS based tracking. We propose a novel way for designing base kernels used in semi-supervised learning. We compute a new set of more "accurate" eigenvectors that are expected to be better aligned to the target.

Chapter 4 addresses the problem that RSS data distribution may not always be static over time, across space or across devices. Most of the current transfer learning methods studies how to make training and testing data have the same distribution in

the input space. In contrast, we consider data distributions in the reproducing kernel Hilbert space. By using the surrogate kernel, we apply an explicit (linear) transform on the Gram matrix of the training data, forcing it to properly "approach" that of the test data, such that the kernel machine learned on the training data generalize well to test domain.

Chapter 5 concludes the thesis.

Chapter 2

Kernel Methods and Nystrom Methods

In this chapter, we will review kernel methods, and related algorithms for regression tasks: (1)Kernel Ridge Regression and (2)Support Vector Regression.

2.1 Kernel Methods

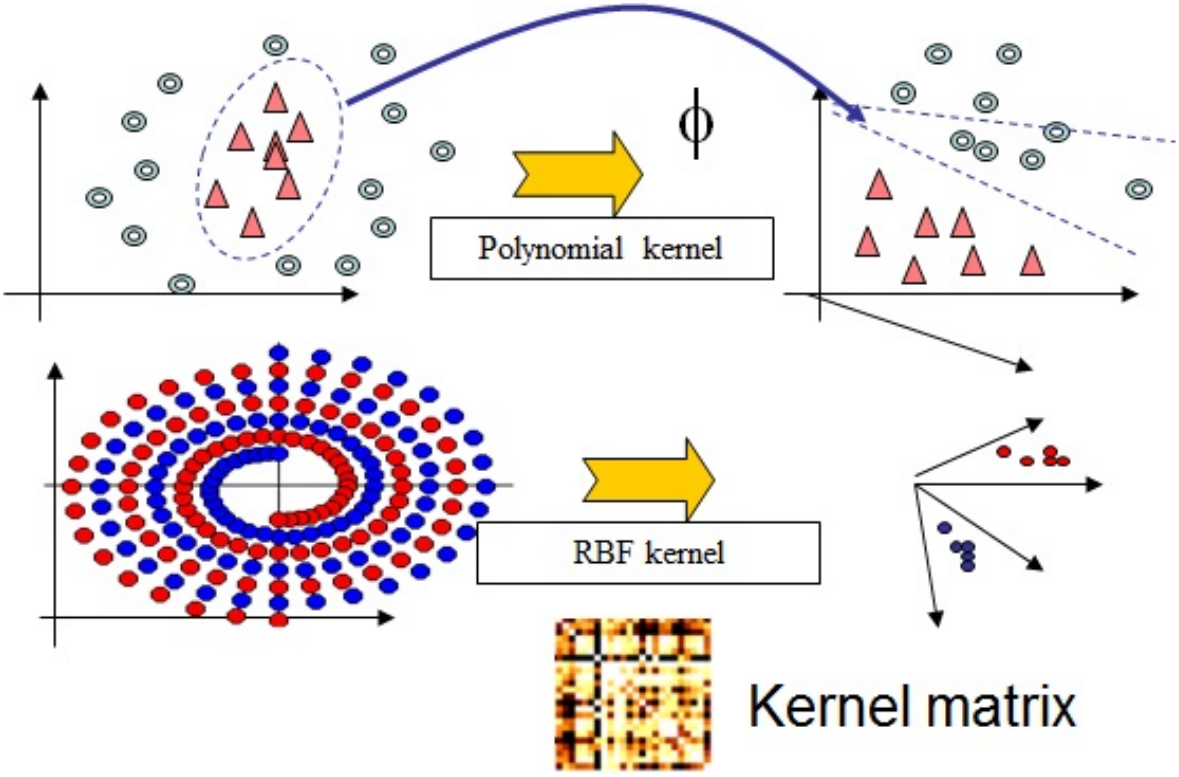


Figure 2.1: the feature map ϕ embeds the data into a feature space where a linear pattern exists.

Over the last decade kernel methods has become popular in machine learning.[67] A kernel method solution includes two components: (1)a module that performs the

mapping into the feature space and (2)a learning algorithm to discover linear patterns in that space, see Figure 2.1. There are two main reasons why this approach should work. On one hand, identifying linear relations has been the focus of machine learning research for decades, and the resulting algorithms are well understood and highly efficient. On the other hand, kernel methods provide a computational shortcut which makes it possible to represent linear patterns in high dimensional spaces to exploit representational power. This shortcut is called a kernel function.

Kernel methods typically doesn't require the coordinates of the embedded points, but only their pairwise inner products. The pairwise inner products can be computed efficiently using a kernel function as

$$k(\mathbf{x}, \mathbf{z}) = \langle \phi(\mathbf{x}), \phi(\mathbf{z}) \rangle$$

where ϕ is called feature map. Given a kernel k, and inputs $\mathbf{x}_1, \mathbf{x}_2, ...\mathbf{x}_n \subseteq X$ the $n \times n$ matrix K, where $K_{ij} = k(\mathbf{x}_i, \mathbf{x}_j)$, is called Gram matrix of k with respect to$\mathbf{x}_1, \mathbf{x}_2, ...\mathbf{x}_n$.

Gram matrix is symmetric. It contains all the information needed to compute the pairwise distances within the data set as shown above. In the Gram matrix there is of course some information that is lost when compared with the original set of vectors. For example the matrix loses information about the orientation of the original data set with respect to the origin, since the matrix of inner products is invariant to rotations about the origin. More importantly the representation loses information about any alignment between the points and the axes. This again follows from the fact that the Gram matrix is rotationally invariant in the sense that any rotation of the coordinate system will leave the matrix of inner products unchanged. We will see that the only information received by the algorithm about the training set comes from the Gram matrix and the associated output values. This observation will characterise all of the kernel algorithms. In this sense we can view the matrix as an information bottleneck that must transmit enough information about the data for the algorithm to be able to perform its task. This view also reinforces the view that the kernel matrix is the central data type of all kernel-based algorithms. All Gram matrices are positive semi-definite.

We can always manipulate and combine simple kernels to obtain more complex and

useful ones. Let k_1 and k_2 be kernels over $X \times X, X \subseteq R^n, a \subseteq R^+, f()$ a real-valued function on $X, \phi : X \to R^N$ with k_3 a kernel over $R^N \times R^N$, and B a symmetric positive semi-definite $n \times n$ matrix, and $p(x)$ is a polynomial with positive coefficients. Then the following functions are valid kernels:

- $k(\mathbf{x}, \mathbf{z}) = k_1(\mathbf{x}, \mathbf{z}) + k_2(\mathbf{x}, \mathbf{z})$
- $k(\mathbf{x}, \mathbf{z}) = ak_1(\mathbf{x}, \mathbf{z})$
- $k(\mathbf{x}, \mathbf{z}) = k_1(\mathbf{x}, \mathbf{z})k_2(\mathbf{x}, \mathbf{z})$
- $k(\mathbf{x}, \mathbf{z}) = f(\mathbf{x})f(\mathbf{z})$
- $k(\mathbf{x}, \mathbf{z}) = k_3(\phi(\mathbf{x}), \phi(\mathbf{z}))$
- $k(\mathbf{x}, \mathbf{z}) = \mathbf{x}'B\mathbf{z}$
- $k(\mathbf{x}, \mathbf{z}) = p(k_1(\mathbf{x}, \mathbf{z}))$
- $k(\mathbf{x}, \mathbf{z}) = exp(k_1(\mathbf{x}, \mathbf{z}))$
- $k(\mathbf{x}, \mathbf{z}) = exp(-\|\mathbf{x} - \mathbf{z}\|^2/(2\sigma^2))$

The last kernel in the list is called Gaussian kernel, they are the most widely used kernels and have been extensively studied in neighbouring fields.[67] The images of all points have norm 1 in the resulting feature space as $k(\mathbf{x}, \mathbf{x}) = exp(0) = 1$. The feature space can be chosen so that the images all lie in a single orthant, since all inner products between mapped points are positive. Note that we are not restricted to using the Euclidean distance in the input space. If for example $k_1(\mathbf{x}, \mathbf{z})$ is a kernel corresponding to a feature mapping ϕ_1 into a feature space F_1, we can create a Gaussian kernel in F_1 by observing that

$$\|\phi_1(\mathbf{x}) - \phi_1(\mathbf{z})\|^2 = k_1(\mathbf{x}, \mathbf{x}) - 2k_1(\mathbf{x}, \mathbf{z}) + k_1(\mathbf{z}, \mathbf{z}),$$

giving the derived Gaussian kernel as

$$k(\mathbf{x}, \mathbf{z}) = exp(-\frac{k_1(\mathbf{x}, \mathbf{x}) - 2k_1(\mathbf{x}, \mathbf{z}) + k_1(\mathbf{z}, \mathbf{z})}{2\sigma^2}),$$

The parameter σ controls the flexibility of the kernel in a similar way to the degree in the polynomial kernel. Small values of σ correspond to large values of degree since, for example, they allow classifiers to fit any labels, hence risking overfitting. In such cases the kernel matrix becomes close to the identity matrix. On the other hand, large values of σ gradually reduce the kernel to a constant function, making it impossible to learn any non-trivial classifier. The feature space has infinite-dimension for every value of σ but for large values the weight decays very fast on the higher-order features. In other words although the rank of the kernel matrix will be full, for all practical purposes the points lie in a low-dimensional subspace of the feature space.

Graph kernels are a family of kernels widely used in semisupervised kernel designs.[67] Graphs pose a twofold challenge: one may both design a kernel on vertices of them and also a kernel between them. In the former case, the graph itself becomes the object defining the metric between the vertices. In the following we discuss kernels on graphs. Denote by $W \subseteq R^{n \times n}$ the adjacency matrix of a graph with $W_{ij} > 0$ if an edge between i, j exists. Moreover, assume for simplicity that the graph is undirected, that is, $W' = W$. Denote by $L = D - W$ the graph Laplacian and by $\hat{L} = I - D^{-1/2} W D^{-1/2}$ the normalized graph Laplacian. Here D is a diagonal matrix with $D_{ii} = \sum_j Wij$ denoting the degree of vertex i. It has been shown that, the second largest eigenvector of L approximately decomposes the graph into two parts according to their sign. The other large eigenvectors partition the graph into correspondingly smaller portions. L arises from the fact that for a function f defined on the vertices of the graph $\sum_{i,j}(f(i) - f(j))^2 = 2f'Lf$. Finally, it shows that, under mild conditions and up to rescaling, L is the only quadratic permutation invariant form which can be obtained as a linear function of W. Hence, it is reasonable to consider kernel matrices K obtained from L. Smola and Kondor suggest kernels $K = r(L)$, which have desirable smoothness properties. Here r is a monotonically decreasing function. Popular choices include (1)diffusion kernel $r(\xi) = exp(-\lambda\xi)$,where $\lambda > 0$ is chosen such as to reflect the amount of diffusion; (2)regularized graph Laplacian $r(\xi) = (\xi + \lambda)^{-1}$, where $\lambda > 0$ is the degree of regularization; (3)p-step random walk $r(\xi) = (\lambda - \xi)^p$ where $\lambda > 0$ is the weighting of steps within a random walk. The function $r(\xi)$ describes the smoothness properties

on the graph and L plays the role of the Laplace operator.

2.2 Kernel Ridge Regression

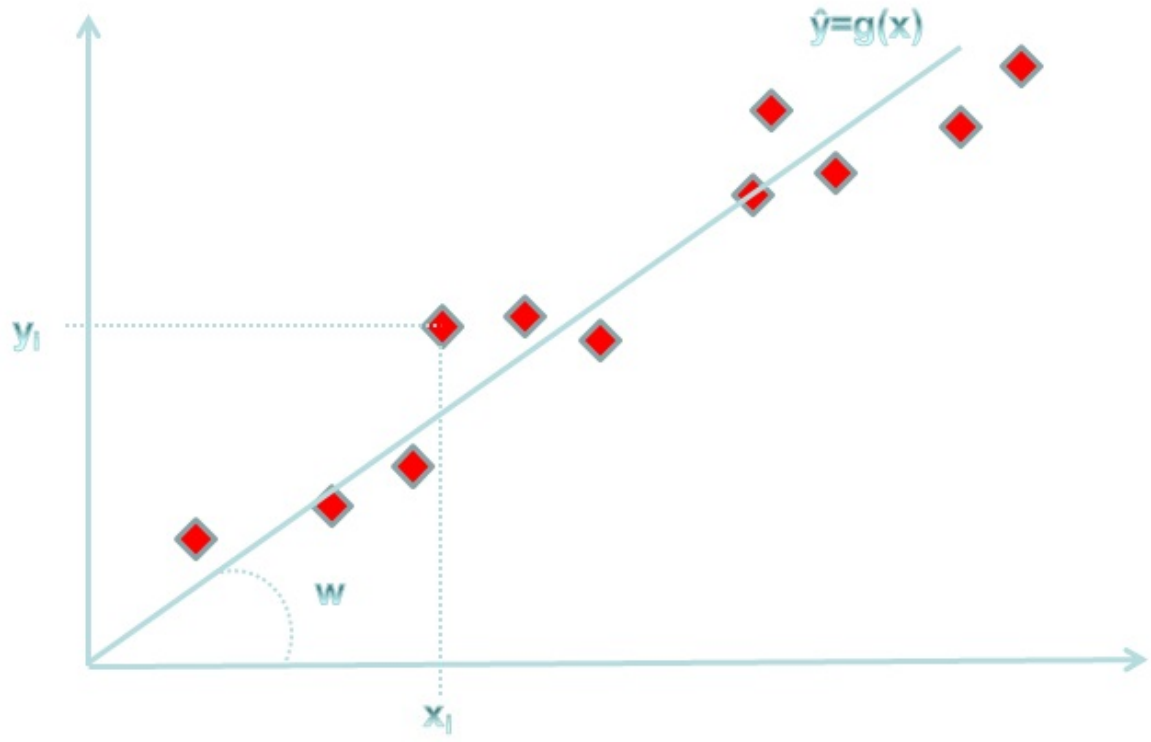


Figure 2.2: A one dimensional linear regression problem.

Having reviewed the background of kernel methods, in the following, we will discuss the technical details of kernel ridge regression. Consider the problem of finding a homogeneous real valued linear function $g(x) = \mathbf{w}'x = \sum_{i=1}^{n} w_i x_i$, that best fits a given training set $S = \{(x_1, y_1), ..., (x_l, y_l)\}$ of points x_i from $\mathcal{X} \subseteq R^n$ with corresponding labels $y_i \subseteq R$. Here we use the notation $\mathbf{x} = (x_1, x_2, ...x_n)$ for the n-dimensional input vectors, while $\mathbf{w}'$ denotes the transpose of the vector $\mathbf{w} \subseteq R^n$. This task is known as linear(ridge) regression [67]. Geometrically it corresponds to fitting a hyperplane through the given n-dimensional points. Figure 2.2 shows an example for $n = 1$.

We use $\xi = y - g(x)$ to denote the error of the linear function on the particular training example. The objective of linear regression is to find a function for which all of training training errors are small. Given a training data set, it can be written in a matrix

form as $\xi = \mathbf{y} - X\mathbf{w}$. The sum of the squares of these errors is the most commonly used measure of discrepancy between the training data and a particular function. Hence, the loss function can be written as $L(\mathbf{w}) = (\xi = \mathbf{y} - X\mathbf{w})'(\xi = \mathbf{y} - X\mathbf{w})$. Linear regression or ridge regression corresponds to solving the following optimization problem

$$\min_{\mathbf{w}} L_\lambda(\mathbf{w}) = \min_{w} \lambda\|\mathbf{w}\|^2 + \|\mathbf{y} - X\mathbf{w}\|^2,$$

After taking the derivative of the cost function with respect to the parameters $\mathbf{w}$ we obtain the equations

$$X'X\mathbf{w} + \lambda\mathbf{w} = (X'X + \lambda\mathbf{I_n})\mathbf{w} = X'\mathbf{y},$$

where $\mathbf{I_n}$ is the $n \times n$ identity matrix. The matrix $(X'X + \lambda\mathbf{I_n})$ is always invertible if $\lambda > 0$, so the solution is given by

$$\mathbf{w} = (X'X + \lambda\mathbf{I_n})^{-1}X'\mathbf{y},$$

Solving this equation for $\mathbf{w}$ involves solving a system of linear equations with n unknowns and n equations. The complexity of this task is $O(n^3)$. The resulting linear function is given by

$$g(\mathbf{x}) = \mathbf{y}'X(X'X + \lambda\mathbf{I_n})^{-1}\mathbf{x}.$$

The equations above computes the weight vector explicitly and is known as the primal solution. Notice that $\mathbf{w}$ can be expressed as a linear combination of the training points, $\mathbf{w} = X'\alpha$. Therefore, we can rewrite the equations above to obtain

$$\mathbf{w} = \lambda^{-1}X'(\mathbf{y} - X\mathbf{w}) = X'\alpha.$$

Hence we have

$$\lambda\alpha = \mathbf{y} - XX'\alpha \Rightarrow \alpha = (XX' + \lambda\mathbf{I}_l)^{-1}\mathbf{y}$$

Solving for α involves solving l linear equations, a task of complexity $O(l^3)$. The resulting prediction function is given by

$$g(\mathbf{x}) = \langle\mathbf{w}, \mathbf{x}\rangle = \sum_i \alpha_i\langle\mathbf{x}_i, \mathbf{x}\rangle = \mathbf{y}'(XX' + \lambda\mathbf{I}_l)^{-1}\mathbf{k},$$

where $k_i = \langle \mathbf{x}_i, \mathbf{x} \rangle$. This is known as the dual solution. The parameter α are know as the dual variables. Denote $G = XX'$, from the dual solution we can see that the information from the training examples is given by the inner products between pairs of training points in G. Similarly, the information about a novel example $\mathbf{x}$ required by the predictive function is just the inner products between the training points and $\mathbf{x}$. The matrix G is referred to as the Gram matrix, which is a $l \times l$ matrix. If the dimension n of the feature space is larger than the number of of training examples l, it becomes more efficient to solve the dual equation. Evaluation of the predictive function in this dual setting is, however, always more costly since the primal involves $O(n)$ operations, while the complexity of the dual is $O(nl)$.

The ridge regression method discussed above addresses the problem of identifying linear relations between the target function and features. Often, however, the relations are indeed nonlinear. To address this problem we will map the data into a new feature space in such a way that the sought relations can be represented in a linear form. Considering an embedding map $\phi(\mathbf{x})$, our objective is to look for a linear relation of the form $\xi = y - g(x) = y - \langle \mathbf{w}, \phi(\mathbf{x}) \rangle$. As shown in the dual solution, all the information the algorithm need is the inner products between data points. So in the feature space it can be represented by $\langle \phi(\mathbf{x}), \phi(\mathbf{z}) \rangle$. The predictive function can be written as $g(\mathbf{x}) = \mathbf{y}'(G + \lambda \mathbf{I}_l)^{-1}\mathbf{k},$, where $G_{ij} = \langle \phi(\mathbf{x}_i), \phi(\mathbf{x}_j) \rangle$ and $k_i = \langle \phi(\mathbf{x}_i), \phi(\mathbf{x}) \rangle$. A kernel is a function k that for all $\mathbf{x}, \mathbf{z} \subseteq X$ satisfies

$$k(\mathbf{x}, \mathbf{z}) = \langle \phi(\mathbf{x}), \phi(\mathbf{z}) \rangle, \tag{2.1}$$

where ϕ is a mapping from X to feature space F.One of the most important properties of Kernel functions is that we can compute the inner product between the embedding of two points in the features space without explicit evaluating their coordinates.

2.3 Support Vector Machine for Regression

Kernel Ridge Regression suffers from the disadvantage that the solution vector $\alpha*$ is not sparse [67]. Hence, to evaluate the learned function on a novel example we must evaluate the kernel with each of the training examples. For large training sets this

will make the response time very slow. In order to encourage sparseness, we need to define a loss function that involves inequalities in its evaluation. This can be achieved by ignoring errors that are smaller than a certain threshold $\epsilon > 0$. For this reason the band around the true output is sometimes referred to as a tube. This type of loss function is referred to as an ϵ-insensitive loss function. Using ϵ-insensitive loss functions leads to the Support Vector Regression algorithms.

The Support Vector algorithm is a nonlinear generalization of the Generalized Portrait algorithm developed in Russia in the sixties. As such, it is firmly grounded in the framework of statistical learning theory, or VC theory, which has been developed over the last three decades by Vapnik and Chervonenkis. In a nutshell, VC theory characterizes properties of learning machines which enable them to generalize well to unseen data. In its present form, the SVM was largely developed at ATT Bell Laboratories by Vapnik and co-workers. Due to this industrial context, SV research has up to date had a sound orientation towards real-world applications. Initial work focused on OCR (optical character recognition). Within a short period of time, SV classifiers became competitive with the best available systems for both OCR and object recognition tasks. A comprehensive tutorial on SV classifiers has been published by Burges in 1998. But also in regression and time series prediction applications, excellent performances were soon obtained. A snapshot of the state of the art in SV learning was recently taken at the annual Neural Information Processing Systems conference in 1999. SV learning has now evolved into an active area of research. Moreover, it is in the process of entering the standard methods toolbox of machine learning. Scholkopf and Smola present a more in-depth overview of SVM regression. Additionally, Cristianini and Shawe-Taylor provide further details on kernels in the context of classification.

Figure 2.3 shows an example of a one-dimensional regression function with an ϵ band. the variables ξ measure the cost of the errors on the training points. These are zero for all points inside the band. The linear ϵ-insensitive loss function is defined by

$$L = max(0, |y - g(\mathbf{x})| - \epsilon), \tag{2.2}$$

where g is a real-valued function on a domain X, $\mathbf{x} \in X$ and $y \in \mathcal{R}$. Similarly, the

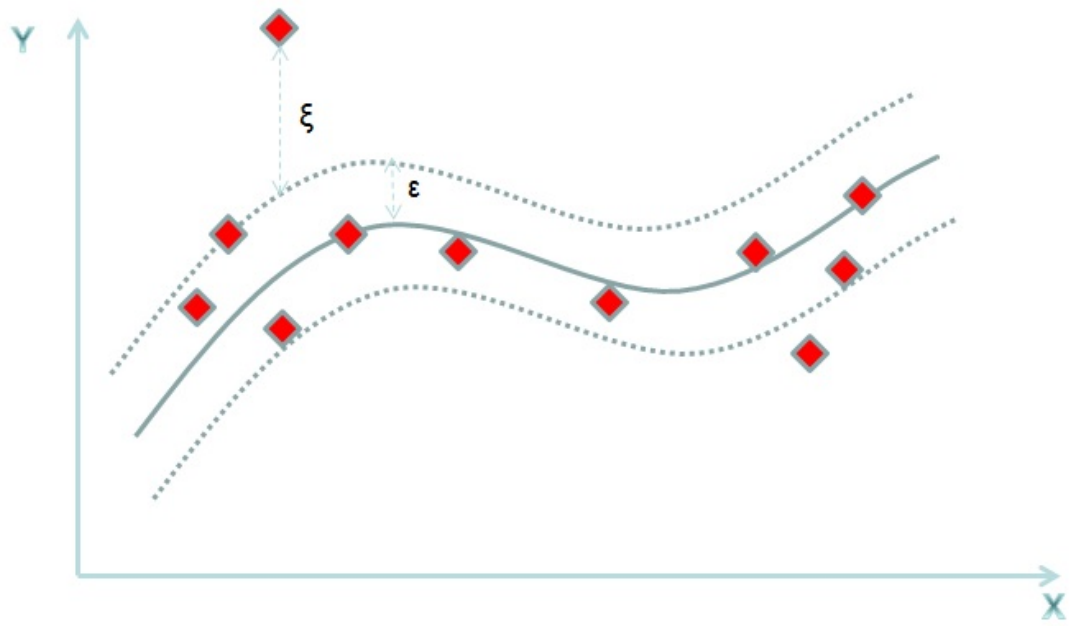


Figure 2.3: Regression using ϵ-insensitive loss function

quadratic ϵ-insensitive loss function is defined by

$$L = max(0, (|y - g(\mathbf{x})| - \epsilon)^2), \tag{2.3}$$

The quadratic ϵ-insensitive support vector regression are defined as

$$\begin{aligned} \min_{\mathbf{w},b,\xi,\hat{\xi}} \quad & \| \mathbf{w} \|^2 + C \sum_{i=1}^{l} (\xi_i^2 + \hat{\xi}_i^2), \\ \textit{subject to} \quad & (\langle \mathbf{w}, \phi(\mathbf{x}_i) \rangle + b) - y_i \leq \epsilon + \xi_i; _{i=1,2,\ldots,l}\,, \\ & y_i - (\langle \mathbf{w}, \phi(\mathbf{x}_i) \rangle + b) \leq \epsilon + \hat{\xi}_i; _{i=1,2,\ldots,l}\,. \end{aligned}$$

We have not constrained the slack variables to be positive since negative values will never arise at the optimal solution. we have further included an offset parameter b that is not penalised. The dual problem can be derived using the standard method and taking into account that $\xi_i \hat{\xi}_i = 0$ and therefore that the same relation $\alpha_i \hat{\alpha}_i = 0$ holds

for the corresponding Lagrange multipliers

$$\begin{aligned} \max_{\alpha,\hat{\alpha}} \quad & \textstyle\sum_{i=1}^{l} y_i(\hat{\alpha}_i - \alpha_i) - \epsilon \sum_{i=1}^{l} (\hat{\alpha}_i + \alpha_i) \\ & -\tfrac{1}{2} \textstyle\sum_{i,j=1}^{l} (\hat{\alpha}_i - \alpha_i)(\hat{\alpha}_j - \alpha_j)(k(\mathbf{x}_i, \mathbf{x}_j) + \tfrac{1}{C}\delta_{ij}), \\ \textit{subject to} \quad & \textstyle\sum_{i=1}^{l} (\hat{\alpha}_i - \alpha_i) = 0, \\ & \hat{\alpha}_i \leq 0, \alpha_i \leq 0; _{i=1,2,\ldots,l} . \end{aligned}$$

Note that by substituting $\beta = \hat{\alpha} - \alpha$ and using the relation $\alpha_i \hat{\alpha}_i = 0$, we can rewrite the dual problem in the following form:

$$\begin{aligned} \max_{\beta} \quad & \textstyle\sum_{i=1}^{l} y_i \beta_i - \epsilon \sum_{i=1}^{l} |\beta_i| - \tfrac{1}{2} \sum_{i,j=1}^{l} \beta_i \beta_j (k(\mathbf{x}_i, \mathbf{x}_j) + \tfrac{1}{C}\delta_{ij}), \\ \textit{subject to} \quad & \textstyle\sum_{i=1}^{l} \beta_i = 0. \end{aligned}$$

Notice that if we set $\epsilon = 0$ we recover the ridge regression, but with an unpenalized offset that give rise to the constraint $\sum_{i=1}^{l} \beta_i = 0$. The predictive function can be written as:

$$f(\mathbf{x}) = \sum_{j=1}^{l} \beta_i^* k(\mathbf{x}_i, \mathbf{x}) + b^*$$

$$b^* = -\epsilon - \frac{\beta_i^*}{C} + y_i - \sum_{j=1}^{l} \beta_i^* k(\mathbf{x}_i, \mathbf{x}_j) \; for \; i \; with \; \beta_i^* > 0$$

If we consider the band of $\pm\epsilon$ around the function output by the learning algorithm, the points that are not strictly inside the tube are tube are support vectors. Those not touching the tube will have the absolute value of the corresponding β_i equal to C; Though the move to the use of the ϵ-insensitive loss was motivated by the desire to introduce sparsity into the solution, remarkably it can also improve the generalization error as measured by the expected value of the squared error in practical experiments.

The quadratic ϵ-insensitive loss follows naturally from the loss function used in ridge regression. There is, however, a linear ϵ-insensitive support vector regression formulated as following:

$$\begin{aligned} \min_{\mathbf{w},b,\xi,\hat{\xi}} \quad & \| \mathbf{w} \|^2 + C \textstyle\sum_{i=1}^{l} (\xi_i + \hat{\xi}_i), \\ \textit{subject to} \quad & (\langle \mathbf{w}, \phi(\mathbf{x}_i) \rangle + b) - y_i \leq \epsilon + \xi_i; _{i=1,2,\ldots,l} , \\ & y_i - (\langle \mathbf{w}, \phi(\mathbf{x}_i) \rangle + b) \leq \epsilon + \hat{\xi}_i; _{i=1,2,\ldots,l} . \end{aligned}$$

The corresponding dual problem can be derived using the standard techniques:

$$\begin{aligned} \max_{\alpha,\hat{\alpha}} \quad & \textstyle\sum_{i=1}^{l} y_i(\hat{\alpha}_i - \alpha_i) - \epsilon \sum_{i=1}^{l} (\hat{\alpha}_i + \alpha_i) \\ & -\tfrac{1}{2} \textstyle\sum_{i,j=1}^{l} (\hat{\alpha}_i - \alpha_i)(\hat{\alpha}_j - \alpha_j) k(\mathbf{x}_i, \mathbf{x}_j), \\ \textit{subject to} \quad & \textstyle\sum_{i=1}^{l} (\hat{\alpha}_i - \alpha_i) = 0, \\ & 0 \leq \hat{\alpha}_i, \alpha_i \leq C; _{i=1,2,\ldots,l} . \end{aligned}$$

Similarly, by substituting $\beta = \hat{\alpha} - \alpha$ and using the relation $\alpha_i \hat{\alpha}_i = 0$, we can obtain the following dual form:

$$\begin{aligned} \max_{\beta} \quad & \textstyle\sum_{i=1}^{l} y_i \beta_i - \epsilon \sum_{i=1}^{l} |\beta_i| - \tfrac{1}{2} \sum_{i,j=1}^{l} \beta_i \beta_j k(\mathbf{x}_i, \mathbf{x}_j), \\ \textit{subject to} \quad & \textstyle\sum_{i=1}^{l} \beta_i = 0, -C \leq \alpha_i \leq C; _{i=1,2,\ldots,l} . \end{aligned}$$

Accordingly, the predictive function can be written as:

$$f(\mathbf{x}) = \sum_{j=1}^{l} \beta_i^* k(\mathbf{x}_i, \mathbf{x}) + b^*$$

$$b^* = -\epsilon + y_i - \sum_{j=1}^{l} \beta_i^* k(\mathbf{x}_i, \mathbf{x}_j) \; for \; i \; with \; 0 < \beta_i^* < C$$

2.4 Nystrom Method

In this work, the Nystrom method plays an critical role in learning kernels. Therefore, we review it in this section. The Nystrom method has been successfully used in speeding up kernel machines [89] [43] and spectral clustering [28]. In [61], it is further shown that several forms of multidimensional scaling [18], including the Landmark MDS [20], FastMap [24] and MetricMap [86], are all variants of the Nystrom method. The Nystrom method is originated from the numerical treatment of the integral equation [5][23]

$$\int p(y) k(x, y) \phi(y) dy = \lambda \phi(x)$$

Here $k(\cdot, \cdot)$ is the kernel function which is usually positive semi-definite, $p(\cdot)$ is the underlying probability density function, and λ and $\phi(\cdot)$ is the eigenvalue and eigenfunction

of the kernel k, respectively. Based on a set of i.i.d. samples $X = \{x_i\}_{i=1}^n$ that are supposed to be drawn from the probability density $p(\cdot)$, the integral can be approximated by

$$\lambda\phi(x) = \int p(y)k(x,y)\phi(y)dy \cong \frac{1}{n}\sum_{j=1}^{n} k(x,x_j)\phi(x_j)$$

When x goes through all x_is in X, we obtain n linear equations:

$$\frac{1}{n}\sum_{j=1}^{n} k(x_i,x_j)\phi(x_j) = \lambda\phi(x_i)$$

which can be written as the eigenvalue decomposition

$$K\phi = n\lambda\phi$$

where $K_{n\times n} = [K_{ij}] = [k(x_i,x_j)]$ is the kernel matrix defined on X, and $\phi = [\phi(x_j)] \in R^n$is the corresponding eigenvector. After solving the equation, evaluation of the eigenfunction $\phi(\cdot)$ at any point x can be achieved as

$$\phi(x) \approx \frac{1}{\lambda n}\sum_{j=1}^{n} k(x,x_j)\phi(x_j)$$

The eigenvalue decomposition scales cubically with the sample size and can be expensive in practice. To reduce the time complexity, one may use only a subset of the available samples, which leads to a much smaller eigenvalue problem. This is commonly known as the Nystrom method and is summarized in Algorithm 1.Note that ϕ_X represents the $n \times 1$ vector formed by evaluating the kernel eigenfunction $\phi(\cdot)$ on the whole sample set $\{x_i\}_{i=1}^n$. It can be used to approximate the eigenvector of the complete kernel matrix after normalization.

Algorithm 1 The Nystrom Method

1. Randomly choose a subset $Z = \{z_i\}_{i=1}^{m}$ from X, and compute the corresponding kernel submatrix $W_{m\times m} = [W_{ij}]$:

$$W_{ij} = k(z_i, z_j)$$

2. Perform the eigenvalue decomposition:

$$W\phi_Z = m\lambda_Z\phi_Z$$

and obtain the corresponding eigenvector $\phi_Z \in R^m$ and eigenvalue $m\lambda_Z$.

3. Compute the interpolation matrix $E_{n\times m} = [E_{ij}]$, where

$$E_{ij} = k(x_i, z_j)$$

4. Extend the eigenvector $\phi_Z \in R^m$ to the whole data set by

$$\phi_X = (m\lambda_Z)^{-1}E\phi_Z$$

Chapter 3

Label-Aware Base Kernels for Indoor WiFi Localization

3.1 Semi-supervised Indoor WiFi Localization

Accurately tracking mobile nodes in wireless sensor networks using radio-signalstrength (RSS) values is a complex and difficult task. Radio signal usually attenuates in a way that is highly nonlinear and uncertain in a complex environment, which may be further corrupted when introducing the mobility of sensor nodes. In the past, many researchers have developed propagation-based algorithms for localizing mobile nodes in a wireless sensor network. These methods [50][65] usually consist of two main steps, by first transforming sensor reading into a distance measure and then recovering the most probable coordinates of sensor nodes. These approaches usually rely on a sophisticated signal propagation model and extensive hardware support. Learning-based approaches [53][44][4] can bypass the ranging process but need relatively more calibration data. However, few of them consider the mobility of sensor nodes. How to estimate the location of mobile nodes when the signal environment is noisy, and when we have much less calibrated (labeled) data and hardware support, is still an open problem.

In this chapter, we address the calibration-effort reduction problem in wifi localization based tracking by proposing a semi-supervised learning approach for learning a set of label aware base kernels. We first collect a small quantity of labelled data at various locations. Then, we train a SVR using label aware base kernels to solve the regression problem, in a semi-supervised manner, using a small amount of labelled data and a large amount of unlabelled data.

3.1.1 Semi-supervised Kernel Design

Among various directions that have been pursued by researchers in semi-supervised learning, semi-supervised kernel design turns to be a promising one because it allows the abundant theories and algorithms in kernel methods to be adopted directly in solving SSL problems. In particular, a large family of algorithms for semi-supervised kernel relies on spectral transformation, where the eigenvectors of the graph Laplacian are used together with the rectified eigenvalues to build the new kernel.

Given an $n \times n$ kernel matrix K, the graph Laplacian is computed as $\mathcal{L} = D - K$, where $D \in \mathcal{R}^{n \times n}$ is a (diagonal) degree matrix such that $D_{ii} = \sum_{j=1}^{n} K_{ij}$. The normalized graph Laplacian is defined as $\tilde{\mathcal{L}} = \mathbf{I} - D^{-1/2} K D^{-1/2}$, where $\mathbf{I}$ is identity matrix. The (normalized) graph Laplacian matrix is positive semi-definite for any function f :

$$f^T L f = \frac{1}{2} \sum_{i,j=1}^{n} K_{ij}(f(i) - f(j))^2 \geq 0.$$

and imposes important smoothness constraints over the graph, which has been widely used in spectral clustering [52], image segmentation [73], and feature selection [32]. Roughly speaking, f is smooth if $f(i) \approx f(j)$ for those pairs with large K_{ij} . This is sometimes informally expressed by saying that f varies slowly over the graph, or that f follows the data manifold. In particular, its smaller eigenvalues correspond to smoother eigenvectors over the graph, i.e., the entries of the eigenvector corresponding to neighboring samples are close to each other. Such smoothness is very useful for predicting the actual class labels. Based on this property, a general principle is applied in spectral transformation to build semi-supervised kernel [77],

$$\tilde{K} = \sum_{i=1}^{n} r(\lambda_i) \phi_i \phi_i^{\top}.$$

Here, λ_i's $(i = 1, 2, ..., n)$ are eigenvalues of the (normalized) graph Laplacian $\mathcal{L} \in \mathcal{R}^{n \times n}$ sorted in an ascending order, ϕ_i's are the corresponding eigenvectors, and $r(\cdot)$ is a non-increasing function which enforces larger penalty for less smooth eigenvectors. The transform $r(\cdot)$ is often chosen from a parametric family, resulting in some familiar kernels. For example

- regularized Laplacian or Gaussian field kernel: $r(\lambda) = \frac{1}{\lambda+\epsilon}$
- diffusion kernel: $r(\lambda) = exp(-\frac{\delta^2}{2}\lambda)$
- one-step random walk or cluster kernel: $r(\lambda) = (\alpha - \lambda)$
- p-step random walk: $r(\lambda) = (\alpha - \lambda)^p$
- inverse cosine: $r(\lambda) = cos(\lambda\pi 4)$
- step function: $r(\lambda) = 1$ if $\lambda \leq \lambda_{cut}$

Each has its own special interpretation. Of course there are many other natural choices for r. Although the general principle is appealing, it does not address the question of which parametric family to use. Moreover, the hyperparameters in a particular parametric family may not suit the task at hand, resulting in overly constrained kernels.

Another group of approaches focus on searching over a nonparametric family of spectral transforms by using convex optimization to maximize kernel alignment to the labeled data. Kernel alignment is a surrogate for classification accuracy, and, importantly, leads to a convex optimization problem. A connection between high alignment and good generalization performance has been established in [19]. The *empirical kernel alignment* [19] evaluates the degree of agreement between a kernel and the learning target, via the use of "ideal kernel" $K^*(\mathbf{x}, \mathbf{z}) = y(\mathbf{x})y(\mathbf{z})$, where $y(\mathbf{x})$ is the target concept (such as the class label chosen from $\{\pm 1\}$ or $\{0, 1\}$ [41]). Given a set of l training examples, corresponding label vector $\mathbf{y} \in \mathcal{R}^{l\times 1}$, and kernel matrix $K \in \mathcal{R}^{l\times l}$, the alignment is computed as

$$A_{K,y} = \frac{\langle K, \mathbf{y}\mathbf{y}^\top \rangle}{l\sqrt{\langle K, K \rangle}},$$

where $\langle K_1, K_2 \rangle = \sum_{ij} K_1(\mathbf{x}_i, \mathbf{x}_j)K_2(\mathbf{x}_i, \mathbf{x}_j)$ is the inner product between matrices. It has been shown that the alignment between kernels is sharply concentrated, i.e., a good alignment on the training set will indicate a good alignment on the test set [19]. On the other hand, $A_{K,y}$ is favorably associated with the generalization performance of a classifier (such as the Parzen window estimator) [19]. Therefore, maximizing the alignment of the kernel with the ideal one provides a general and effective way for kernel

design. In this work, we adopt a different alignment criterion between two kernels K and K' from [17],

$$\rho(K, K') = \frac{\langle K_c, K'_c \rangle_F}{\|K_c\|_F \|K'_c\|_F},$$

where K_c is the centralized version of K.

This criterion provides a novel concentration bound, and shows the existence of good predictors for kernels with high alignment, in both classification and regression tasks.

The concept of ideal kernel and its implications have led to several successful methods for kernel learning. The common theme of these methods is to use eigenvectors of the kernel matrix to span a set of base kernels, and then optimize the weighting in the combined kernel via maximizing its alignment with the target (or ideal kernel). For example,

- [19] proposed to compute the weighting of each base kernel proportional to the inner product of the corresponding eigenvector with the target.
- [76] presented a framework for computing sparse combination of the base kernels.
- [17] showed that the weighting in the maximal alignment kernel can be solved via quadratic programming.
- In [42], a semi-definite programming formulation was adopted to learn a kernel matrix $\tilde{K}$ that is maximally aligned with the ideal kernel.
- In [100], an order constraint on the transformed eigenvalue is further considered. The order constraint reflects important prior belief that smoother eigenvectors should be given higher priority in building the kernel.
- [16]proposed the algorithm based on the concept of local Rademacher complexity, which is upper-bounded by tailsum of the eigenvalues of kernels. The authors proposed a regularization formulation for controlling the tailsum instead of the traditional way on restricting trace norm of kernels.

Note that the base kernels are not necessarily orthogonal eigenvectors computed from one empirical kernel matrix (or its Laplacian matrix). In many cases the base kernels themselves can be different empirical kernel matrices that are either from different domains/views of the data, or simply computed by varying the kernel parameters, such as [42, 17, 16]. Most of the algorithms we have reviewed here apply to both cases.

3.1.2 Motivation of Our Approach Comparing to Kernel Based Methods

The majority of the methods in the literatures for kernel matrix design learning belongs to the wide category of kernel methods, since they all focus on how to learn a good kernel matrix by revising the eigen-spectrum of the kernel matrix. However, note that one limitation of this family of algorithms is that the base kernels spanned by kernel eigenvectors can have low quality because they are computed regardless of the label. Therefore, we may not expect that the kernel eigen-structures faithfully reflects the target variable. To alleviate this problem, we propose to compute a set of desired "eigenvectors" via extrapolation of the ideal kernel eigenfunction to build label-aware base kernels. The biggest difference between our approach and existing SSL (kernel) methods is that our approach utilizes the given labels to compute a set of more "accurate" eigenvectors to span the base kernels. In other words, our approach not only learns a better eigen-spectrum but more importantly, a better set of eigenvectors. Therefore, our approach can also be deemed as a generalization of the existing SSL kernel learning methods.

3.2 Designing Label-Aware Base Kernels

Kernel target alignment is an important criterion widely used in semi-supervised kernel design [19]. A higher alignment with the ideal kernel indicates the existence of a good classifier with a higher probability [19][17]. It can be easily observed that the overall alignment of the mixed kernel depends directly on the individual base kernel alignment scores. For example, it has been shown that the optimized alignment between a kernel

$\tilde{K} = \sum_{i=1}^{k} \beta_i v_i v_i^\top$ and the ideal kernel $yy^\top$ is $\tilde{A}(y) = \frac{1}{k}\sqrt{\sum_{i=1}^{k} \langle v_i, y\rangle_F^4}$, where $\langle v_i, y\rangle$ is the alignment for the ith eigenvector. However, in practice, the base kernels spanned by the eigenvectors of the kernel matrix might deviate a lot from the target due to various practical factors, such as noise, choice of kernel types/parameters, or the difficulty of the classification problem.

In the following, we consider building more "accurate" eigenvectors to span better base kernels. Note that one reason of the low quality of the base kernels spanned by kernel eigenvectors is that they are computed regardless of the label. Therefore, we may not expect that the kernel eigen-structures faithfully reflects the target variable. To alleviate this problem, we propose to compute a set of desired "eigenvectors" via extrapolation of the ideal kernel eigenfunction. We first discuss the connection between kernel eigenvectors and class labels, and then introduce the concept of kernel eigenfunction extrapolation to build label-aware kernel eigenvectors.

3.2.1 Kernel Eigenvectors and Class Labels

Proposition 1 *Given l labeled examples ordered from c classes, with ideal kernel in the form of*

$$K^* = \begin{bmatrix} 11'_{l_1} & 0 & \dots & 0 \\ 0 & 11'_{l_2} & \cdots & 0 \\ \vdots & \dots & \ddots & \vdots \\ 0 & \dots & 0 & 11'_{l_c} \end{bmatrix}. \tag{3.1}$$

where l_i is size of the ith class. Let $Y \in \mathcal{R}^{l\times c}$ be the class label, i.e., $Y_{ij} = 1$ if $\mathbf{x}_i$ is in class j; and $Y_{ij} = 0$ otherwise. Then the ith non-zero eigenvector of K^* is $\frac{1}{\sqrt{l_i}}Y_i$, where Y_i is the ith column of Y.

Proposition 1 *Let the eigenvalue decomposition of K^* be $K^*\mathbf{v}^* = \lambda^*\mathbf{v}^*$. Since K^* only has c different rows (orthogonal to each other), it has rank c with $n-c$ zero eigenvalues. Note that the ith entry of $\mathbf{v}^*$ equals $\frac{1}{\lambda^*}K^*(i,:)\mathbf{v}^*$, and K^* has a block-wise constant structure. Therefore $\mathbf{v}^*$ is piecewise constant. Write $\mathbf{v}^*$ as $[\underbrace{v_1,...,v_1}_{l_1}\underbrace{v_2,...,v_2}_{l_2},...,\underbrace{v_c,...,v_c}_{l_c}]'$. Then the eigensystem can be written as an equation group $m_k v_k = \lambda^* v_k$ for $k =$*

$1, 2, ..., c$. *Each equation in it leads to two conditions:* $\lambda^* = l_k$, *or* $v_k = 0$. *However, it's impossible to set* $\lambda^* = l_k$ *for* $k = 1, 2, ..., C$, *since the size of different classes can be different. The only feasible way is to set* λ^* *equal to one of the* ml_k*'s, i.e.,* $\lambda^* = l_{k_0}$, *and at the same time set* $v_k = 0$ *for all the* $k \neq k_0$. *There are* c *different ways to choose* k_0, *i.e.,* $k_0 = 1, 2, ..., c$. *For each choice of* k_0, *the eigenvalue is* $\lambda^* = l_{k_0}$; *as to the eigenvector, all its entries corresponding to class* k $(k \neq k_0)$ *will be zero, and the entries corresponding to class* k_0 *will be* $\frac{1}{\sqrt{l_{k_0}}}$ *(since they are equal and should normalize to 1). This completes the proof.*

Proposition 1 shows that non-zero eigenvectors of the ideal kernel correspond exactly to the classes labels (up to a scaling). For example, [73] shows that the eigenvectors corresponding to the second smallest eigenvalue of the normalized graph Laplacian provides a relaxed solution for a two-class clustering problem[1].

Therefore, eigenvectors and class labels have intrinsic connections. The main difference is that eigenvectors of the kernel matrix can be noisy and may fail to reveal underlying cluster structures due to their unsupervised nature; in comparison, class label represents prior knowledge and is always a clean, piecewise constant vector. The connection indicates that if we can expand "ideal" kernel eigenvectors from labeled samples to the whole data set and obtain a set of high-quality eigenvectors that align better to class labels, then the resultant base kernels will also have a higher target alignment. To achieve this goal, we need notions of eigenfunction and its extrapolation via the Nystrom extension.

3.2.2 Eigenfunction Expansion

Let $\mathcal{A}$ be a linear operator on a function space. The eigenfunction f of $\mathcal{A}$ is any non-zero function that returns itself from the operator, i.e., $\mathcal{A}f = \lambda f$, where λ is the eigenvalue. In this work, we are interested in the case where $\mathcal{A}$ is a symmetric, positive semi-definite kernel $K(\mathbf{x}, \mathbf{z})$. The corresponding eigenfunction $\phi(\cdot)$, given the underlying

[1]Positive entries in this eigenvector will be deemed as positive class and negative entries will be indicative of the negative class.

sample distribution $p(\mathbf{x})$, is defined as [88]

$$\int K(\mathbf{x},\mathbf{z})\phi(\mathbf{x})p(\mathbf{x})d\mathbf{x} = \lambda\phi(\mathbf{z}). \tag{3.2}$$

The standard numerical method to approximate the eigenfunctions and eigenvalues in Eq.(3.2) is to replace the integral with the empirical average [28, 88]

$$\int K(\mathbf{x},\mathbf{z})p(\mathbf{x})\phi(\mathbf{x})d\mathbf{x} \approx \frac{1}{q}\sum_{i=1}^{q} K(\mathbf{x}_i,\mathbf{z})\phi(\mathbf{x}_i), \tag{3.3}$$

where $\mathbf{x}_{i,\ i=1,2,...,q}$ is drawn from the distribution $f(\cdot)$. By choosing $\mathbf{z}$ as $\mathbf{z} = \mathbf{x}_{i,\ i=1,2,...,q}$, Eq.(3.3) extends to a matrix eigenvalue decomposition $K\mathbf{v} = \lambda\mathbf{v}$, where K is the kernel matrix defined as $K_{ij} = K(\mathbf{x}_i,\mathbf{x}_j)$ for $1 \le i,j \le q$, and $\mathbf{v}$ is the discrete counterpart of ϕ in that $\phi(\mathbf{x}_i) \approx \mathbf{v}(i)$. Then the eigenfunction can be extended by

$$\phi(\mathbf{z}) \approx \frac{1}{q\lambda}\sum_{i=1}^{q} K(\mathbf{z},\mathbf{x}_i)\mathbf{v}(i). \tag{3.4}$$

This is known as the Nystrom extension [89], which means that the eigenvectors of the empirical kernel matrix evaluated on a finite sample set can be used as approximators to the whole eigenfunction of the linear operator. Interestingly, Eq.(3.4) is proportional to the projection of a test point computed in kernel PCA [10]. The approximation can be justified by examining the convergence of eigenvalues and eigenvectors as the number of examples increases [68, 10].

3.2.3 Extrapolating Ideal Kernel Eigenfunctions

Motivated by the eigenfunction extension, we propose to extrapolate the ideal kernel eigenvectors as follows. Suppose we are given the labeled set $X_l = \{\mathbf{x}_i\}_{i=1}^{l}$ with labels $Y \in \mathcal{R}^{l\times c}$, where c is the number of classes, and the unlabeled set $X_u = \{\mathbf{x}_i\}_{i=l+1}^{n}$. Then, in order to expand the ideal kernel eigenfunction from X_l to the whole data set $X_l \cup X_u$, we can choose $\{\mathbf{x}_i\}_{i=1}^{q}$ in (3.4) as X_l, choose $\mathbf{z}$ in (3.4) as $X_l \cup X_u$, and choose $\mathbf{v}(i)$ as the labels of X_l. Suppose the estimated kernel eigenvectors are denoted as $u_k \in \mathcal{R}^{n\times 1}$ for $k = 1,2,...,c$, corresponding to the c classes, then we have

$$u_k(i) = \frac{1}{l\lambda_k}\sum_{\mathbf{x}_j\in X_l} K(\mathbf{x}_i,\mathbf{x}_j)Y_{jk}. \tag{3.5}$$

Here, λ_k is the eigenvalue corresponding to the kth class, which according to Proposition 1 is proportional to the size of the kth class. To guarantee that the estimated labels/eigenvector entries are in a reasonable range, one can also normalize the weighting coefficients $K(\mathbf{x}_i, \mathbf{x}_j)$ by $\sum_j K(\mathbf{x}_i, \mathbf{x}_j)$.

The advantage of extrapolating the ideal kernel eigenfunction is that the resultant eigenvector incorporates label information directly. Therefore, empirically they typically have higher alignment with the target compared with the eigenvectors of the kernel matrix, the computation of the latter being totally irrespective of available class labels. With such label-aware eigenvectors, we will then have better base kernels for semi-supervised learning.

3.2.4 Combining Base Kernels

Having obtained a set of extrapolated ideal kernel eigenvectors, we can use them to span base kernels for semi-supervised kernel design. In case the number of labeled sample is very limited, using label-aware eigenvectors alone may not be sufficient. Therefore, it's safer to incorporate the kernel eigenvectors as well. Suppose we have obtained a set of c extrapolated eigenvectors u_1,u_2,..,u_c, as well as a set of k eigenvectors v_1, v_2, ..., v_k, from the kernel matrix (or graph Laplacian). Then we want to learn the following kernel

$$\tilde{K} = \sum_{i=1}^{c} \alpha_i u_i u_i^\top + \sum_{j=1}^{k} \beta_j v_j v_j^\top . \tag{3.6}$$

The mixing coefficients can be determined by maximizing the alignment to the target. In other words, it will be automatically determined which parts take higher weights. If the problem is easy and kernel eigenvectors already are accurate enough, then they will play a major role in shaping the new kernel; on the other hand, if the kernel eigenvectors turn out to be noisy and poorly aligned to the target, then the label-aware eigenvectors will probably assume higher weights. In the literature, there are various ways to compute the weights such as uniform weighting, independent alignment-based weighting, or the quadratic programming approach. In this work, we adopt a well-known method

alignf [17] and a very recent method **local rademacher complexity** [16] that determines the mixture weights jointly by seeking to maximize the alignment between the convex combination kernel and the target kernel,

With the learned kernel $\tilde{K}$, one can use $\tilde{K}$ as the similarity matrix and plug it in SVM for training and testing.

The whole algorithm is summarized in Algorithm 2.

Algorithm 2 Input: labeled samples $X_l = \{\mathbf{x}_i\}_{i=1}^{l}$, unlabeled sample set $X_u = \{\mathbf{x}_i\}_{i=l+1}^{n}$; Gaussian Kernel $k(\cdot,\cdot)$, label $Y = [\mathbf{y}_1, \mathbf{y}_2, ..., \mathbf{y}_c] \in \mathcal{R}^{l\times c}$.

---

1. Compute the kernel matrix defined among $X_l \cup X_u$ and X_l, as $K_{nl} \in \mathcal{R}^{n\times l}$; compute the degree matrix $D_n = diag(K_{nl} \cdot 1_{l\times 1})$;
2. Perform eigenfunction extrapolation as $[u_1, u_2, ..., u_c] = D_n^{-1} K_{nl} Y$;
3. Use the Nystrom method [28] to compute eigenvectors corresponding to dominant k eigenvalues of kernel matrix or diminishing k eigenvalues of the (normalized) graph Laplacian, as $[v_1, v_2, ..., v_k]$;
4. Compute the weights of the base eigenvectors $[u_1, u_2, ..., u_c, v_1, v_2, ..., v_k]$;
5. Compute the new kernel $\tilde{K} = \sum_{i=1}^{c} \alpha_i u_i u_i^\top + \sum_{j=1}^{k} \beta_j v_j v_j^\top$;
6. Apply kernel $\tilde{K}$ in SVM for training and testing.

3.2.5 Multiple Kernel Setting

In previous section, we only consider the use of a single (empirical) kernel matrix K to construct base kernels in semi-supervised kernel learning. Recently, researchers have emphasized the need to consider multiple kernels that may correspond to heterogenous data sources (or views) and can improve the model flexibility. In case no physically meaningful multiple domains exist, one can always artificially create them. For example, by changing the kernel width parameter, multiple RBF kernel matrices can be constructed [17, 16, 42]. Then these different empirical kernel matrices can all be used to construct base kernels (or themselves can be directly used as base kernels) for ultimate kernel learning.

In this section, we incorporate this idea in our method. Suppose we have a number

of p different kernel matrices (or graph Laplacians), corresponding to p different sources. Then, we will compute both the unsupervised kernel eigenvectors (u_i's, $i = 1, 2, ..., c$) as well as the label-aware kernel eigenvectors (v_j's, $j = 1, 2, ..., k$) for each kernel. Ultimately, all these eigenvectors are fed together into a multiple kernel learning procedure. More specifically, we can write the final kernel as,

$$\tilde{K} = \sum_{t=1}^{p} (\sum_{i=1}^{c} \alpha_{ti} u_{ti} u_{ti}^{\top} + \sum_{j=1}^{k} \beta_{tj} v_{tj} v_{tj}^{\top}) \tag{3.7}$$

Next, altogether $p(k+c)$ base kernels can be fed into a kernel learning procedure such as **alignf** procedure [17] or **local rademacher complexity** [16] to determine the mixing weights.

3.2.6 Complexity

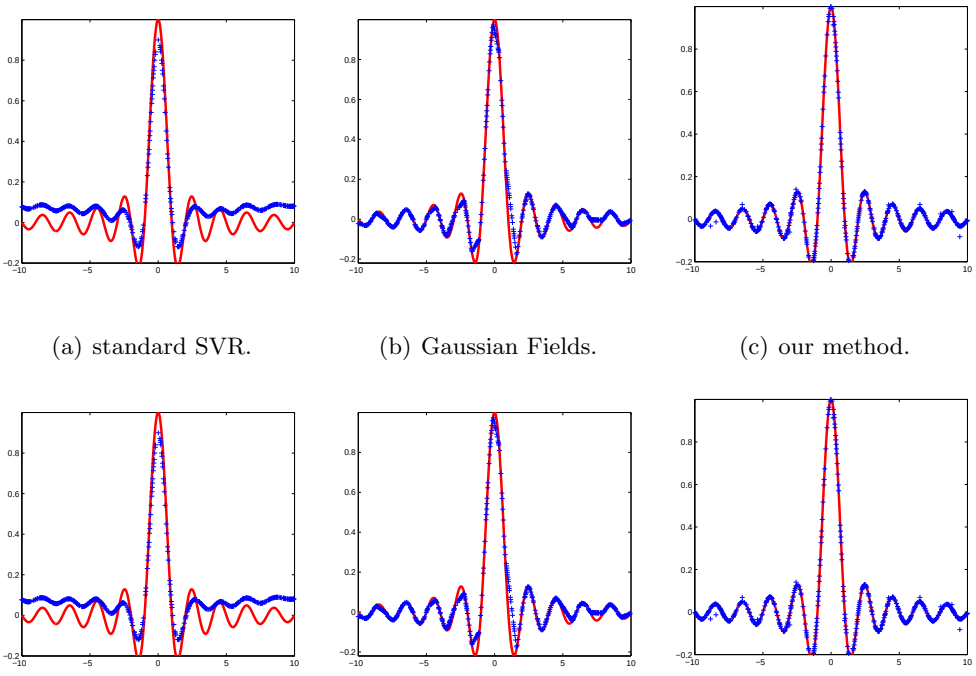
In Algorithm 1, step 1 and 2 takes $O(nc)$ time and space, where n is the sample size and c the number of classes; step 3 takes $O(np^2)$ time and $O(np)$ space; step 4 takes $O(lc)$ time and space; in applying the learned kernel $\tilde{K}$ in SVM, we only need the $l \times l$ block of $\tilde{K}$ corresponding to labeled samples, and the $u \times l$ block corresponding to the block between unlabeled and labeled samples. Therefore, the space needed is $O(nl)$. Step 5 takes $O(cl)$ time. In step 6, the training takes empirically $O(l^{2.3})$ time using the libsvm package, and testing takes $O(pn + ln)$, and the time complexity is $O(nl + np^2 + l^{2.3})$. In practice, we have $l, p \ll n$. Therefore, overall our algorithm has a linear time and space complexities.

3.3 Experiments

3.3.1 Regression

In this section, we report empirical results of our algorithm in a regression task using synthetic data generated by $y = sinc(x)$. We generate 1000 points using the sinc function, and randomly choose 10% as labeled data, and 90% as unlabeled data. We compare our result with (1) cluster kernel [13] (2) diffusion kernel [38]; (3) maximal

alignment kernel [19]; and(4) Gaussian Fields based method-winner method in the contest [99] and standard support vector regression (SVR) [78]. We use the Gaussian kernel in the experiments. For the kernel width, we first compute b_0 as the inverse of the average squared pairwise distances, and then choose b among $b_0 \cdot \{\frac{1}{50}, \frac{1}{25}, \frac{1}{10}, \frac{1}{5}, 1, 5, 10\}$ that gives the best performance. We set $\epsilon = 0.01$ in the support vector regression setting. The regularization parameter C is chosen as $\{0.1, 1, 10, 100, 1000, 10000\}$. In Figure 3.1, we plot the regression results. Here, red points are the true values, and blue points are estimated ones. As can be seen, our approach provides better regression results compared with Gaussian Fields based method. We use the square root of the mean squared error to measure the regression quality The error of standard SVR is 0.0854; that of Gaussian Fields based method is 0.0123; while ours is only around 0.0052.

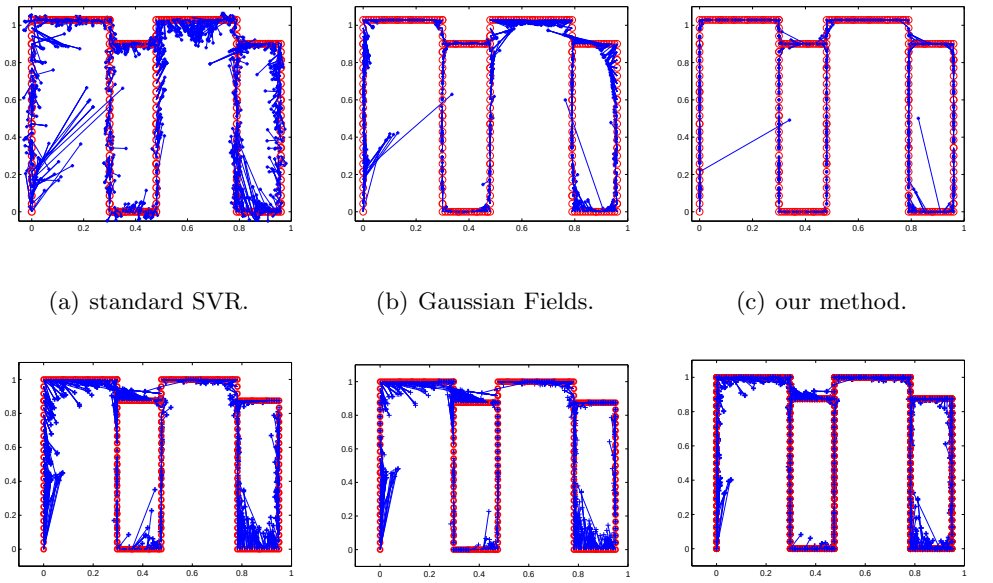


(a) standard SVR. (b) Gaussian Fields. (c) our method.

(d) Cluster Kernel. (e) Diffusion Kernel. (f) Maximal Alignment Kernel.

Figure 3.1: Regression results by different methods. For each test point, red color represents the true value and blue color represents the estimation.

3.3.2 Indoor WiFi Localization



(a) standard SVR. (b) Gaussian Fields. (c) our method.

(d) Cluster Kernel. (e) Diffusion Kernel. (f) Maximal Alignment Kernel.

Figure 3.2: Localization results by different methods. For each test point, a line is connected between the true and the estimated location/coordinates.

In this section, we report empirical results of our algorithm in indoor location estimation using received signal strength (RSS) that a client device received from Wi-Fi access points [91]. The data are published in ICDM2007 semi-supervised learning contest. We compare our result with the following state of the art semi-supervised kernel design methods: (1) cluster kernel [13] (2) diffusion kernel [38]; (3) maximal alignment kernel [19]; and(4) Gaussian Fields based method-winner method in the contest [99]. In particular, we adopt the support vector regression (SVR) [78] that works on the learned kernels on semi-supervised kernel design algorithms. We normalize the labels y_i's such that they scale in the range $[0, 1]$. We use the Gaussian kernel in the experiments. In semi-supervised learning parameter selection is an open problem. In this work, the parameters are chosen as follows. For the kernel width, we first compute b_0 as the inverse of the average squared pairwise distances, and then choose b among $b_0 \cdot \{\frac{1}{50}, \frac{1}{25}, \frac{1}{10}, \frac{1}{5}, 1, 5, 10\}$ that gives the best performance. For all methods, we set

$\epsilon = 0.05$ in the support vector regression setting. The regularization parameter C is chosen as $\{0.1, 1, 10, 100, 1000, 10000\}$. Cluster kernel method uses $10\%n$ diminishing eigenvectors from the normalized graph Laplacian; other methods use the top 10% eigenvectors of the kernel matrix.

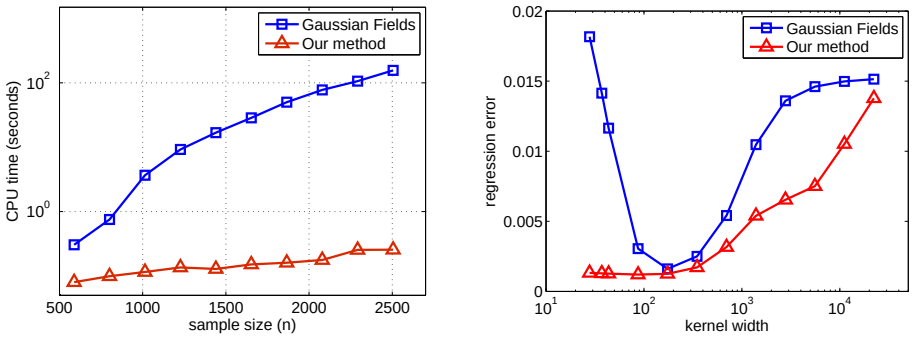
In Figure 3.2, we plot the regression results on the 2-D plane. Here, red circles are the true coordinates, and blue dots are estimated ones. A line is connected between every pair of true and estimated points. As can be seen, our approach provides better localization results compared with other baseline methods. We use the square root of the mean squared error to measure the regression quality. The error of standard SVR is 2.5×10^{-3}(Figure 3.2(a)); that of Gaussian Fields based method is 1.61×10^{-3}(Figure 3.2(b)); while ours is only $\mathbf{1.19} \times 10^{-3}$(Figure 3.2(c)). Traditional semi-supervised kernel design methods performs only slightly better than standard SVR, as can been see that cluster kernel method is 2.11×10^{-3}(Figure 3.2(d)); diffusion kernel method is 1.97×10^{-3}(Figure 3.2(e)); and maximal aligned kernel method is 1.92×10^{-3}(Figure 3.2(f));. Our regression error is reduced by about 25% compared with Gaussian Fields based method, more than 30% with other state of the art semisupervised kernel design methods, and more than 50% compared with the standard supervised SVR. The results are listed in Table 3.1

Table 3.1: comparison of the wifi localization error rate using different semi-supervised kernel methods.

SVR	Cluster Kernel	Diffusion Kernel	Maximal Aligned Kernel	Gaussian Fields Method	Ours
$2.5\times10^{-3}\pm0.0002$	$2.11\times10^{-3}\pm0.0009$	$1.97\times10^{-3}\pm0.0003$	$1.92\times10^{-3}\pm0.0002$	$1.61\times10^{-3}\pm0.0005$	$\mathbf{1.19}\times10^{-3}\pm0.0008$

In Figure 3.3(a), we gradually increase the number of unlabeled samples from 200 to $2,000$, and examine the time consumption. As can be seen, our approach is orders of magnitudes' faster compared with Gaussian Fields based method.In Figure 3.3(b), we plot the regression error of the two methods with regard to the Gaussian kernel width. As can be seen, our approach is less sensitive to the choice of the kernel parameters. This makes it a practical in real-world applications. From this example, we can see that semi-supervised kernel design can give competitive performance compared with

stat-of-the-art SSL algorithms that focus on estimating the labels (but not learning a kernel). This validates the importance of a good kernel in semi-supervised learning tasks. Of course, there are many SSL algorithms whose focus is not on learning kernel. We choose the Gaussian Fields based method as an example for comparison because it has shown to provide stat-of-the-art results in this localization task [91].



(a) Time versus sample size.

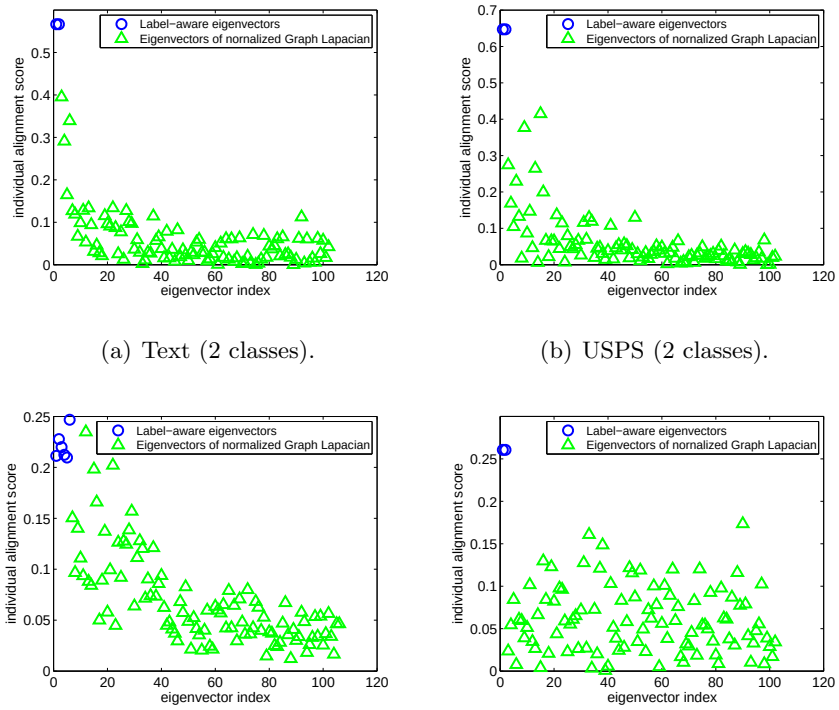
(b) Error versus kernel width.

Figure 3.3: Properties of the Gaussian Fields based method and our approach.

3.3.3 Classification

In this section, we compare the following semi-supervised kernel design methods: (1) cluster kernel [13], where $r(\cdot)$ is chosen as linear function $r(\lambda) = \lambda$; (2) diffusion kernel $r(\lambda) = \exp(-\lambda/\delta)$ [38]; (3) maximal alignment kernel [19] using the top 0.1n eigenvectors from the kernel matrix; (4) our approach; (5) non-parametric graph kernel [100] using the first $p = 0.1n$ eigenvectors from the normalized Laplacian $\tilde{\mathcal{L}}$. Evaluation is based on the alignment on the unlabeled data, and classification error of SVM using the learned kernel. Here, for a fair comparison, our method only uses one empirical kernel matrix instead of multiple empirical kernel matrix altogether.

We used the Gaussian kernel $K(\mathbf{x}_1, \mathbf{x}_2) = \exp(-\|\mathbf{x}_1 - \mathbf{x}_2\|^2 \cdot b)$ in all our experiments. In semi-supervised learning parameter selection is an open problem. In this work, the parameters are chosen as follows. For the kernel width, we first compute b_0 as the inverse of the average squared pairwise distances, and then choose b among $b_0 \cdot \{\frac{1}{50}, \frac{1}{25}, \frac{1}{10}, \frac{1}{5}, 1, 5, 10\}$ that gives the best performance. The parameter δ and ϵ are



(a) Text (2 classes).

(b) USPS (2 classes).

(c) Coil (6 classes).

(d) BCI (2 classes).

Figure 3.4: The individual target alignment score of label-aware base eigenvectors and the traditional kernel eigenvectors **on the unlabeled data**. For simplicity of visualization, here the reported score is the average alignment between one eigenvector and all the c target variables/classes.

Table 3.2: Classification performance using different semi-supervised kernel design schemes. For each cell, the top row is the mean/std of the kernel alignment score (in $[0, 1]$) on the test set, and in bracket is the averaged time consumption (in seconds); the bottom row is the mean/std of classification error (%).

Data size/dim	Spectral Graph kernel	Ours	Cluster kernel linear	Diffusion kernel	Max-alignment kernel
Digit1	0.29±0.07 (84.9)	0.82±0.02 (1.2)	0.13±0.005 (2.4)	0.10±0.001 (13.0)	0.14±0.001 (12.6)
1500×241	4.31±1.93	4.89± 0.85	5.37±1.23	6.13±1.63	**3.82±1.23**
USPS	0.23±0.08 (74.9)	0.66±0.04 (1.2)	0.43±0.001 (2.5)	0.06±0.001 (16.0)	0.06±0.01 (12.7)
1500×241	7.47± 4.41	**6.64±1.27**	**6.56±1.02**	7.27±0.59	9.81±0.49
COIL2	0.11±0.005 (73.4)	0.55±0.07 (1.2)	0.10±0.001 (2.4)	0.05±0.003 (8.4)	0.07±0.00 (5.3)
1500×241	18.49±2.47	**13.44±2.41**	18.51±4.66	19.08±2.05	19.32±1.89
BCI	0.07±0.003 (9.9)	0.14±0.04 (0.4)	0.04±0.001 (0.2)	0.07±0.003 (0.4)	0.07±0.002 (0.5)
400× 241	**32.95±3.38**	**32.99±3.10**	42.02±2.89	33.58±2.83	34.85±2.75
COIL	0.01±0.001 (199.5)	0.11±0.05 (0.4)	0.08±0.002 (2.58)	0.06±0.001 (8.3)	0.07±0.001 (5.5)
1500×241	21.90±3.24	**9.14±0.96**	10.89±1.12	11.67±1.43	11.75±1.49
g241n	0.40±0.003 (108.2)	0.33±0.03 (1.4)	0.03±0.007 (2.5)	0.04±0.00 (20.3)	0.04±0.00 (6.7)
1500×241	**13.64±1.28**	24.11±1.73	26.59±3.96	19.68±1.52	18.61±1.75
Text	0.13±0.01 (181.0)	0.30±0.02 (20.1)	0.03±0.001 (68.1)	0.03±0.00 (208.0)	0.03±0.004 (130.7)
1500×11960	25.55±1.65	**23.42±1.46**	32.90±6.64	24.89±1.81	26.78±4.88
usps38	0.48±0.004 (77.3)	0.84±0.02 (1.2)	0.12±0.001 (1.6)	0.11±0.001 (6.8)	0.11±0.001 (4.5)
1200×256	4.82±1.33	**2.82±0.83**	5.10±0.89	6.06±1.01	6.06±0.85
usps49	0.40±0.13 (82.1)	0.86±0.01 (1.2)	0.09±0.001 (1.9)	0.08±0.001 (9.3)	0.07±0.001 (8.9)
1296×256	2.83±0.92	**1.98±0.52**	6.29±2.11	8.26±0.83	10.67±1.24
usps56	0.48±0.06 (80.0)	0.86±0.01 (1.2)	0.12±0.001 (1.7)	0.09±0.003 (18.2)	0.11±0.001 (5.0)
1220×256	2.87±0.92	**2.44±0.59**	3.89±1.57	3.85±0.97	5.79±1.06
usps27	0.58±0.004 (101.8)	0.91±0.06 (1.2)	0.37±0.001 (2.3)	0.10±0.001 (11.8)	0.13±0.001 (6.9)
1376×256	1.79±0.42	**1.21±0.25**	1.80±0.25	2.28±0.56	4.80±1.29
odd/even	0.21±0.008 (419.0)	0.65±0.03 (1.6)	0.12±0.001 (8.8)	0.03±0.004 (38.5)	0.08±0.00 (22.3)
2007×256	10.14±2.11	**9.58±1.56**	14.59±1.49	14.08±2.04	15.64±2.91

chosen from $\{10^{-5}, 10^{-3}, 10^{-1}, 1\}$. Each algorithm is repeated 30 times with 50 labeled samples randomly chosen for each class. Cluster kernel method and non-parametric graph kernel method use $10\%n$ diminishing eigenvectors from the normalized graph Laplacian; other methods use the top 10% eigenvectors of the kernel matrix. Results are reported in Table 3.2. As can be seen, our algorithm gives competitive performance and at the same time very efficient.

3.3.4 Evaluation of Superiority of Label-aware Base Kernels

In previous sections, we have demonstrated that the proposed method outperforms existing methods in semi-supervised kernel learning tasks including classification and regression. To further verify that this superiority is attributed to the newly proposed label-aware base kernels, in this section, we examine in more detail the usefulness of the proposed label-aware base kernels in two different ways. First, we examine the individual alignment score of the base kernels as an index of their quality. Second, we apply state-of-the-art kernel combination schemes [17, 16] on two sets of base kernels, one is the traditional (unsupervised) base kernels, and the other includes the label-aware base kernel, and compare their learning performance. We call this side-by-side comparison. This comparison clearly shows that by adding the newly proposed base kernels, the learning performance of the mixed kernel improves.

Alignment Score Comparison

In Figure 3.4, we examine alignment score of the label-aware eigenvectors (blue circles) and those from the normalized Graph Laplacian[2]. Here, the reported score is the average alignment between one eigenvector and all the c target variables. As can be seen, the label-aware eigenvectors almost always have higher or at least very similar alignment scores compared with the eigenvectors of the graph Laplacian.

[2]Empirically, eigenvectors from the normalized graph Laplacian have higher target alignment than those from the kernel matrix.

Table 3.3: Side-by-side comparison of the learning performance using single empirical kernel matrix.

Data size/dim	alignf (baseline)	alignf (ours)	local rademacher complexity(baseline)	local rademacher complexity(ours)
Digit1	0.27±0.02 (0.8)	0.82±0.02 (1.2)	0.49±0.02(16.9)	0.82±0.02(17.1)
1500×241	10.73±0.54	4.89±0.85	6.45±0.63	5.18±0.56
USPS	0.02±0.01 (0.5)	0.66±0.04 (1.2)	0.03±0.01 (6.5)	0.72±0.01(5.7)
1500×241	14.15±1.92	6.64±1.27	10.09±1.49	8.18±1.26
COIL2	0.15±0.07 (0.6)	0.55±0.07 (1.2)	0.18±0.01(13.2)	0.51±0.03 (11.3)
1500×241	19.92±2.14	13.44±2.41	18.54±3.43	14.22±2.12
BCI	0.01±0.04 (0.6)	0.14±0.04 (0.4)	0.06±0.01(17.2)	0.18±0.04(14.3)
400×241	51.36±3.50	32.99±3.10	39.36±3.35	33.09±2.59
COIL	0.09±0.01 (0.3)	0.11±0.05 (0.4)	0.08±0.01 (20.3)	0.12±0.05 (20.4)
1500×241	15.92±2.31	9.14±0.96	19.94±4.32	10.65±4.33
g241n	0.31±0.01 (0.6)	0.33±0.03 (1.4)	0.20±0.02(14.5)	0.33±0.01(17.8)
1500×241	29.03±0.67	24.11±1.73	26.35±1.16	23.90±1.16
Text	0.13±0.02 (18.8)	0.30±0.02 (20.1)	0.14±0.03(25.6)	0.34±0.02(22.1)
1500×11960	30.18±2.28	23.42±1.46	27.34±2.87	23.09±2.21
usps38	0.53±0.01 (0.9)	0.84±0.02 (1.2)	0.56±0.02(25.6)	0.85±0.03(23.6)
1200×256	6.25±0.80	2.82±0.83	4.63±0.41	2.54±0.32
usps49	0.52±0.02 (0.7)	0.86±0.01 (1.2)	0.37±0.01(25.0)	0.87±0.02(20.9)
1296×256	5.70±0.77	1.98±0.52	4.63±0.24	2.09±0.39
usps56	0.52±0.02(0.7)	0.86±0.01 (1.2)	0.55±0.03(39.4)	0.86±0.05(22.5)
1220×256	4.09±0.72	2.44±0.59	5.54±0.52	2.36±0.53
usps27	0.56±0.01 (0.8)	0.91±0.06 (1.2)	0.78±0.02(5.1)	0.88±0.01(19.1)
1376×256	1.98±0.16	1.21±0.25	2.09±0.31	1.18±0.27
odd/even	0.22±0.01 (0.8)	0.65±0.03 (1.6)	0.31±0.02(27.0)	0.70±0.02(14.1)
2007×256	13.91±1.52	9.58±1.56	10.27±0.64	8.82±0.84

Table 3.4: Side-by-side comparison of the learning performance using multiple empirical kernel matrix.

Data size/dim	alignf (baseline)	alignf (ours)	local rademacher complexity(baseline)	local rademacher complexity(ours)
Digit1	0.82±0.02 (1.2)	0.86±0.02(2.8)	0.82±0.02(17.1)	0.79±0.01(30.8)
1500×241	4.89±0.85	4.70±0.92	5.18±0.56	4.54±0.61
USPS	0.66±0.04 (1.2)	0.72±0.04(2.7)	0.72±0.01(5.7)	0.73±0.01(15.1)
1500×241	6.64±1.27	6.09±1.02	8.18±1.26	7.04±1.04
COIL2	0.55±0.07 (1.2)	0.58±0.02 (3.2)	0.51±0.03 (11.3)	0.53±0.04 (34.1)
1500×241	13.44±2.41	13.14±2.66	14.22±2.12	14.02±2.66
BCI	0.14±0.04 (0.4)	0.22±0.04(1.4)	0.18±0.04(14.3)	0.22±0.04(25.7)
400×241	32.99±3.10	30.27±2.02	33.09±2.59	30.45±1.88
COIL	0.11±0.05 (0.4)	0.13±0.01 (2.1)	0.12±0.05 (20.4)	0.15±0.03 (42.3)
1500×241	9.14±0.96	9.04±0.98	10.65±4.33	10.61±4.81
g241n	0.33±0.03 (1.4)	0.37±0.03(5.6)	0.33±0.01(17.8)	0.34±0.01(24.5)
1500×241	24.11±1.73	22.36±1.93	23.90±1.16	23.09±1.01
Text	0.30±0.02 (20.1)	0.33±0.02(23.8)	0.34±0.02(22.1)	0.34±0.01(46.3)
1500×11960	23.42±1.46	23.36±0.96	23.09±2.21	23.00±1.13
usps38	0.84±0.02 (1.2)	0.81±0.02(2.7)	0.85±0.03(23.6)	0.84±0.02(54.7)
1200×256	2.82±0.83	2.63±0.33	2.54±0.32	2.27±0.45
usps49	0.86±0.01 (1.2)	0.86±0.01(2.9)	0.87±0.02(20.9)	0.87±0.01(23.7)
1296×256	1.98±0.52	1.81±0.22	2.09±0.39	1.90±0.45
usps56	0.86±0.01 (1.2)	0.87±0.01(1.6)	0.86±0.05(22.5)	0.85±0.01(44.7)
1220×256	2.44±0.59	2.35±0.59	2.36±0.53	2.27±0.32
usps27	0.91±0.06 (1.2)	0.90±0.01(1.9)	0.88±0.01(19.1)	0.90±0.01(35.6)
1376×256	1.21±0.25	0.72±0.21	1.18±0.27	1.00±0.19
odd/even	0.65±0.03 (1.6)	0.67±0.02(3.1)	0.60±0.02(14.1)	0.67±0.02(21.9)
2007×256	9.58±1.56	8.90±1.43	10.36±0.84	9.36±0.84

Side-by-Side Learning Performance Comparison

In this section, we perform a side-by-side comparison among two sets of base kernels to examine their performance in kernel learning. The first set only contains unsupervised kernel eigenvectors; the second set further adds label-aware base kernels. For both settings, we use Gaussian kernels $K(\mathbf{x}_1, \mathbf{x}_2) = \exp(-\|\mathbf{x}_1 - \mathbf{x}_2\|^2 \cdot b)$ as base kernels. For each base kernel, we extract p eigenvectors from its graph Laplacian, where p is chosen among $0.01n, 0.05n, 0.1n, 0.15n, 0.2n$. Here, algnf [17]and local rademacher [16] are chosen as the baseline kernel combination schemes.

The kernel parameter and regularization parameter C in SVM are the same as Section 3.3.3. In the local rademacher complexity method, parameters θ and C are chosen among $\{0.1n, 0.5n, 0.8n, 0.9n, n\}$ and $\{0.1, 1, 10, 100, 1000, 10000\}$. We examine the use of a single empirical kernel matrix (see Table 3.3), as well as a set of multiple empirical kernel matrices whose kernel width parameters vary in $b_0 \cdot \{\frac{1}{50}\frac{1}{25}, \frac{1}{10}, \frac{1}{5}, 1, 5, 10\}$ (see Table 3.4). As can be seen, by incorporating the label-aware base kernels, the learning performances are improved in most of the data sets in both kernel combination schemes. We also note that the improvement is less significant in case multiple empirical kernel matrices are used. We speculate that when more empirical kernel matrices are used (with varying kernel parameter), they may have higher chances to align to the target. However, in some data sets, such as usps38,usps49 and etc, although multiple empirical kernel matrices are used which lead to more base kernels, their performances still can be significantly improved after adding the label aware base kernels. We speculate that these data sets are very difficult, and even under a wider choice of the kernel parameters, the resultant unsupervised empirical kernel matrices are still poor candidate for base kernels, and have to rely on using the label information to further improve their quality. This clearly demonstrate the usefulness of the proposed label-aware base kernels in difficult learning tasks.

Chapter 4

Surrogate Kernels for Indoor WiFi Localization

4.1 Transfer Learning In Indoor WiFi Localization

Most machine-learning based Indoor WiFi Localization methods rely on collecting a lot of labeled data to train an accurate localization model offline for use online, and assuming that the distributions of RSS data over different time periods are static. However, it is expensive to calibrate a localization model in a large environment. Moreover, the RSS values are noisy and can vary with time [53][44]. As a result, even in the same environment, the RSS data collected in one time period may differ from those collected in another. How can we build machine learning models that are robust to the change of domains or environment? How to fully exploit the training labels obtained in older domains so as to save the human labors in the localization tasks in the new environment? Transfer learning provides an effective tool for solving these problems.

Transfer learning aims to extract the knowledge from one or more source domains and applies the knowledge to a different, but related target domain. It has drawn considerable attention in the areas of machine learning, and there has been a large amount of methods proposed in the literatures. Among the various formulations, covariate shift is probably the most popular one and has been studied extensively. In statistics, a covariate is a variable which is possibly predictive of the outcome under study. It is also termed as explanatory variable, independent variable, or predictor in some scenarios. Covariate shift refers to the situation that the marginal distribution $P_S(x)$ of the source domain S differs from $P_T(x)$, the marginal distribution of the target domain T for evaluation, while the conditional distribution $P_S(y|x) = P_T(y|x)$ is shared between the two domains.

Early works in solving the covariate shift problem include [83, 11], which treated

different domains as tasks and applied the multi-task learning. Daumé III and Marcu [35] investigated how to train a general model with data from both a source domain and a target domain for domain adaptation in natural language processing tasks. Recently, several research work [74, 96, 34, 80] has converged along the direction of estimating a point-wise re-weighting on the training data to minimize the generalization error in testing. For example, Huang *et al.* [34] applied the *kernel mean matching* (KMM) to account for the distribution difference, such that the means of the training and test points in a reproducing kernel Hilbert space (RKHS) are close. Sugiyama *et al.* [80, 79] proposed a framework to estimates the importance ratio which is simultaneously equipped with model selection. Their idea is to find an importance estimate $\hat{w}(\mathbf{x})$ such that the Kullback-Leibler divergence from the true test input density $P_{te}(\mathbf{x})$ to its estimate $\hat{P}_{te}(\mathbf{x}) = \hat{w}(\mathbf{x})P_{tr}(\mathbf{x})$ is minimized. In [59], instead of learning point-wise re-weighting coefficients, the authors proposed to learn the so called transfer components by minimizing the Maximum Mean Discrepancy (MMD) criterion defined in the RKHS, which measures the distance between distributions of two samples. The transfer components are in the form of pre-parameterized empirical kernel maps and can handle out-of-samples conveniently.

4.1.1 Domain Adaptation

In the computer vision community, similar problems also have drawn considerable interest from researchers, which is called *domain adaptation*. In the following we make a review of these methods and categorize them into several classes. Note that some of the related methods may not be directly applicable to the wifi-localization problem, due to the need of labeled training samples in the target domain.

- Metric learning based approaches. [64] proposed a modified metric learning approach that learns a symmetric transformation matrix to account for the mismatch between two domains. The transformation matrix is required to be symmetric may be overly restrictive for some applications, which requires the dimensionality of the two domains are equal. It was later extended to an asymmetric version of

with a different regularizer [40] . These approaches require a few labels from the target domain and are proposed specifically for classification problem. However, in the wifi localization problem, the labels are the location coordinates in the floor plan which leads to a more challenging, regression problem. In addition, we do not have any label in the testing domain; but only labels in the training domain;

- Low-rank methods. The low-rank structure has been shown to be effective to map the data to some low-dimensional spaces which minimize the statistical difference between two domains. In [6] , the authors learned a subspace projection such that the projected data are similarly distributed across the domains. They minimize the MMD between the two domains using Gaussian RBF and polynomial kernels. An intra-class scatter term is included to take care of the discriminative information. Similarly, this method is not suited for regression problem. Instead of the MMD criterion,[7] proposed to compute the data distribution via kernel density estimation, then learn a subspace projection via minimizing the Hillenger Distance on the statistical manifold. They showed better experimental results than using the counterpart MMD. However, the resulting problem is nonconvex and inefficient to solve for large-scale data. [26] found a mapping matrix between two domains by aligning their respective subspaces obtained via PCA. Although the objective function considered in this method is the difference between the sample mean in the feature space, it is used as an indicator of the distance between two distributions in the input space.

- Sparse coding methods. Sparse coding is another popular type of feature learning approaches in domain adaptation, one would like to learn the representations of both domains such that the domain discrepancy is minimized under the new representation. [49] introduced the MMD regularizers to learn new representations in the Graph Regularized Sparse Coding framework. A regularizer similar to the objective function of SVM, is introduced in [70], therefore labels from both domains are required for these methods. [87] presented a semi-coupled dictionary learning (SCDL) model, it assumes the sparse coefficients from one domain to

be identical to those observed at the other domain via a linear projection. [33] extended it to a coupled dictionary learning approach such that these two coupled dictionaries are aligned simultaneously. One big disadvantage of these methods is that the optimization problem involved has to be solved alternatively.

- Phantom domain methods. Another line of approaches model the domain shift from the source domain to the target domain by some intermediate phantom domains. The source and the target domains represent by two subspaces, respectively. One can then find a number of intermediate subspaces interpolating between those two [30]. Since the new representations have a very high dimensionality, a dimension reduction step is required before actually training the classifiers. [29] and [97] both developed efficient kernel methods to harvest an infinite number of intermediate phantom domains. An extension was proposed in [75], the authors model each class of the source domain by a subspace, which is then associated with a subspace from the target domain. The Karcher mean of source domain subspaces and that of target domain ones are used to compute a geodesic direction. The domain shift is then modeled by the parallel transportations from all the source subspaces to their corresponding target subspaces. Instead of finding intermediate subspaces on the Grassmannian manifold, [54] proposed iteratively synthesizing intermediate subspaces in a manner which gradually reduces the reconstruction residue of the target data. However, it is still an open problem of choosing the best sampling strategy for intermediate domains. Choosing the number of subspaces to sample and the dimensionality of the subspaces, and handling the high dimensionality of the new representations are also challenging.

- Deep learning methods. Most recently, deep learning has been applied in the domain adaptation problems and has achieved impressive performance in computer vision [21, 55]. So far these approaches mainly rely on a sufficiently large and labeled dataset from the source domain. However, it remains questionable how the "deep" methods could be utilized when there are limited training data from the source domain.

4.1.2 Motivation of Our Approach

The existing methods study how to make training and testing data have the same distribution in the input space. In contract, few attempts has been made specifically to cater to kernel methods, where being considered should be the data distributions in the reproducing kernel Hilbert space (RKHS). While in general one may want to consider data distribution in the input space, the behavior of kernel methods are determined in a more complex mechanism due to the interplay between the kernel and the data distribution. In particular, the kernel methods work by applying a linear algorithm in the kernel-induced feature space, where the algorithm performance depends directly on the data distribution in this space. Therefore, we believe that making the training and testing data have similar distributions in the feature space will be a more direct way in tackling the covariate shift problem for kernel-based learning algorithms. In this chapter, we address the cross domain indoor WiFi localization problem by proposing a transfer learning approach for learning a surrogate kernel between the source domain and the target domain. By using the surrogate kernel, we can apply an explicit (linear) transform on the Gram matrix of the training data, forcing it to properly align that of the test data, such that the kernel machine learned on the training data generalize well to test domain.

Superiority of our method comparing to existing domain adaptation methods.

- our method does not require labels in the target domain; in comparison, metric learning based methods require labels in both source and target domain;
- our method aligns source and target domain in the kernel space, which is a non-linear mapping; in comparison, low rank based methods align the two domains in input space and are usually linear, which can be less flexible;
- out method has a global optimal solution; in comparison, existing sparse coding based method are usually non-convex and can only obtain a local optimal solution;

- our method doesn't need to handle high dimensional space as introduced by phantom domains method;
- our method doesn't rely on large data set for training; in comparison, deep learning methods have to require a huge number of labelled samples which are typically non-trivial to obtain in WiFi-localization applications.

4.2 Designing Surrogate Kernels

The kernel matrix describes important pairwise similarity among input patterns, and is the basic building block of the kernel machine. Given a kernel matrix K defined on a set of input samples $\mathcal{X}$, it implicitly induces a non-linear, possibly infinite dimensional mapping $\mathcal{X} \to \varphi(\mathcal{X})$, where the inner product $\langle \varphi(\mathbf{x}), \varphi(\mathbf{y}) \rangle$ can be conveniently estimated as $K(\mathbf{x}, \mathbf{y})$ for $\mathbf{x}, \mathbf{y} \in \mathcal{X}$. In other words, the kernel matrix K can be deemed as the inner product $\langle \varphi(\mathcal{X}), \varphi(\mathcal{X}) \rangle$. Therefore, if two kernel matrices K_1 and K_2 from different domains are close to each other (the definition of closeness will be made clear in later section), then their corresponding feature map $\varphi(\mathcal{X}_1)$ and $\varphi(\mathcal{X}_2)$ will also be close, and as a result the distributions of the data in the kernel-induced feature space will be close, too (the former is a necessary condition for the latter). In other words, matching the data distribution in the feature space can be equivalently and conveniently cast as manipulating the kernel matrix, which avoids the difficulty of handling the possibly infinite-dimensional feature vectors $\varphi(\mathbf{x})$'s.

Inspired by this simple observation, we propose to transform the kernel matrix in the source domain such that it is more similar to that in the target domain. By doing this, the feature map (RKHS) embodied via the kernel matrices will be similar for the two domains, allowing models trained in one domain to generalize well to the other. However, the kernel matrix is data dependent. Given kernel matrices defined on two different data sets, it is difficult to evaluate the closeness between them since they are of different sizes and their row/column correspondence is undefined, not to mention rectifying one to align to the other.

To solve this problem, in this work we propose the concept of *surrogate kernel*. More

specifically, suppose we have a kernel matrix $K_{\mathcal{X}}$ defined on a data set $\mathcal{X}$. On the other hand, we are given a new data set $\mathcal{Z}$. Here, we want to generate a surrogate kernel of $K_{\mathcal{X}}$ by somehow "projecting" it from $\mathcal{X}$ to $\mathcal{Z}$, denoted by $K^{Z\leftarrow X}$. The surrogate kernel $K^{X\leftarrow Z}$ should inherit key structures of $K_{\mathcal{X}}$ but, instead of being defined on $\mathcal{X}$, $K^{X\leftarrow Z}$ is defined on the new set $\mathcal{Z}$. Therefore, $K^{X\leftarrow Z}$ can be used in replacement of $K_{\mathcal{X}}$ when we want to compare $K_{\mathcal{X}}$ with any kernel matrice defined on $\mathcal{Z}$. In order to define the structure of the kernel matrix and how to faithfully preserve it across domains, we will resort to the following theorem.

Theorem 1 *(Mercer) Let $K(\mathbf{x}, \boldsymbol{y})$ be a continuous symmetric non-negative function which is positive definite and square integrable w.r.t. the distribution $p(\cdot)$, then*

$$K(\mathbf{x}, \boldsymbol{y}) = \sum_{i=1}^{\infty} \lambda_i \phi_i(\mathbf{x}) \phi_i(\boldsymbol{y}). \tag{4.1}$$

Here the non-negative eigenvalues λ_i's and the orthonormal eigenfunctions ϕ_i's are the solutions of the following integral equation

$$\int K(\mathbf{x}, \boldsymbol{y}) p(\boldsymbol{y}) \phi_i(\boldsymbol{y}) d\boldsymbol{y} = \lambda_i \phi_i(\boldsymbol{y}). \tag{4.2}$$

The Mercer's theorem [66] is the fundamental theorem underlying the reproducing kernel Hilbert space. It states that any psd kernel can be reconstructed by the kernel eigenfunctions (4.1). In particular, given data set $\mathcal{X}$ with distribution $p(\cdot)$ and corresponding kernel matrix $K_{\mathcal{X}}$, if we can compute the kernel eigenspectrum λ_i's and continuous eigenfunctions $\phi_i(\cdot)$'s (4.2), we will then be able to freely evaluate the kernel (4.1) on arbitrary pair of point. In particular, if the evaluation is performed on a new data set $\mathcal{Z}$, a regenerated kernel matrix on $\mathcal{Z}$ will then be obtained. This regenerated kernel matrix builds entirely on the eigensystem conveyed via the original kernel matrix $K_{\mathcal{X}}$, therefore we believe it preserves key structure of $K_{\mathcal{X}}$ and can be used as its surrogate on the new domain $\mathcal{Z}$.

Next comes the problem of estimating the eigen-spectrum and continuous eigenfunctions of $\mathcal{X}$, which are solutions of the integral equation (4.2). Thanks to [88, 69], they can be approximated asymptotically by a finite-sample eigenvalue-decomposition

on the empirical kernel matrix $K_{\mathcal{X}}$. In the following we derive a concrete approximation. Suppose that we have a set of samples $\mathcal{X} = \{\mathbf{x}_i\}_{i=1}^n$ drawn from $p(\cdot)$. Then we can approximate the integral in (4.2) by the empirical average:

$$\frac{1}{n}\sum_{j=1}^{n} k(\mathbf{x}, \mathbf{x}_j)\phi_i(\mathbf{x}_j) \simeq \lambda_i \phi_i(\mathbf{x}). \tag{4.3}$$

Choosing $\mathbf{x}$ in (4.3) from $\mathcal{X}$ leads to a standard eigenvalue decomposition $K_{\mathcal{X}}\Phi_{\mathcal{X}} = \Phi_{\mathcal{X}}\Lambda_{\mathcal{X}}$, where $K_{\mathcal{X}}(i,j) = k(\mathbf{x}_i, \mathbf{x}_j)$, $\Phi_{\mathcal{X}} \in \mathcal{R}^{n\times n}$ has orthonormal columns and $\Lambda_{\mathcal{X}} \in \mathcal{R}^{n\times n}$ is a diagonal matrix. The eigenfunctions $\phi_i(\cdot)$'s and eigenvalues λ_i's in (4.2) can be approximated respectively by columns of Φ and diagonal entries of Λ, up to a scaling constant. According to (4.3), the eigenfunction $\phi_i(\mathbf{x})$ at any point $\mathbf{x}$ can be extrapolated by $\phi_i(\mathbf{x}) = \frac{1}{\lambda_i n}\sum_{j=1}^{n} k(\mathbf{x}, \mathbf{x}_j)\phi_i(\mathbf{x}_j)$. Therefore, if we want to evaluate the eigenfunctions $\phi_i(\cdot)$'s $(i = 1, ..., n)$ on the set $\mathcal{Z}$, we can write them in matrix form as

$$\Phi_{\mathcal{Z}} = K_{\mathcal{Z}\mathcal{X}}\Phi_{\mathcal{X}}\Lambda_{\mathcal{X}}^{-1}, \tag{4.4}$$

where $K_{\mathcal{Z}\mathcal{X}}$ is the cross similarity between $\mathcal{Z}$ and $\mathcal{X}$, evaluated using kernel k.

We provide an illustrative example in Figure 4.1. Here, $\mathcal{X}$ (+) is drawn from a Gaussian distribution, and we computed a kernel matrix $K_{\mathcal{X}}$ using RBF kernel. We plot its 2nd largest eigenvector (also marked with +). We also computed the (continuous) eigenfunction corresponding to this eigenvector, $\phi_2(u)$ (solid curve). Next, we draw data $\mathcal{Z}$ ($\circ$) from a uniform distribution, and we evaluate the eigenvector $\phi_2(u)$ on $\mathcal{Z}$ (also marked with $\circ$). As can be seen, both eigenvectors (defined on $\mathcal{X}$ and $\mathcal{Z}$) lie on the eigenfunction curve. However, $\mathcal{X}$ and $\mathcal{Z}$ play quite different roles: $\mathcal{X}$ is the source of information, based on which the kernel eigenfunction is estimated; in comparison, $\mathcal{Z}$ is only a set of points that passively receives information from $\mathcal{X}$the source domain on which the eigenfunction is evaluated. In other words, $\mathcal{Z}$ only passively receives the eigen-information created by $\mathcal{X}$.

Based on this observation, we can make Question 1 more explicit as: how to reconstruct $K^{\mathcal{Z}\leftarrow\mathcal{X}}$ on $\mathcal{Z}$ such that it maximally reflects the information of $K_{\mathcal{X}}$ in terms of eigenvalues and eigenfunctions?

Answer 1 *A natural solution is to reconstruct a surrogate kernel $K^{\mathcal{Z}\leftarrow\mathcal{X}}$ using all the*

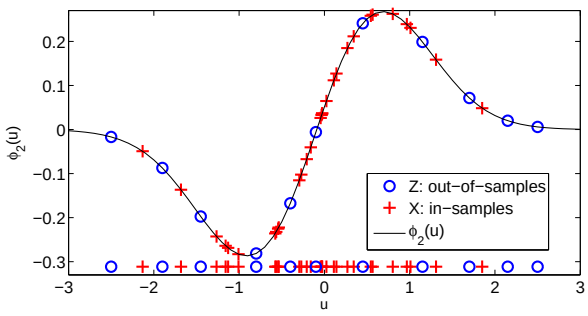


Figure 4.1: Extrapolating the kernel eigenfunction $\phi_2(u)$ from in-sample set $\mathcal{X}$ to out-of-sample set $\mathcal{Z}$.

eigenvalues of $\mathcal{X}$*, but its re-extrapolated eigenvectors on* $\mathcal{Z}$*, i.e.,*

$$K^{\mathcal{Z}\leftarrow\mathcal{X}} = \Phi_{\mathcal{Z}}\Lambda_{\mathcal{X}}\Phi'_{\mathcal{Z}} = \left(K_{\mathcal{Z}\mathcal{X}}\Phi_{\mathcal{X}}\Lambda_{\mathcal{X}}^{-1}\right)\Lambda_{\mathcal{X}}\left(K_{\mathcal{Z}\mathcal{X}}\Phi_{\mathcal{X}}\Lambda_{\mathcal{X}}^{-1}\right)' = K_{\mathcal{Z}\mathcal{X}}K_{\mathcal{X}}^{-1}K_{\mathcal{X}\mathcal{Z}}. \tag{4.5}$$

Interestingly, equation (4.5) coincides with the matrix completion version of the Nystrom method [89]. We call $K^{\mathcal{Z}\leftarrow\mathcal{X}}$ the *surrogate kernel* of $K_{\mathcal{X}}$ on data $\mathcal{Z}$.

We note that in [84], a kernel extrapolation method is proposed to extend a given kernel matrix to unseen data. However, the kernel extrapolation is used to prevent expensive kernel construction (say, via SDP programming) on large data, and the feature map $\mathbf{x} \to \mathcal{R}^n$ needs to be given; while the surrogate kernel is used as a bridge to solve the distribution difference problem, and the non-linear feature mapping is handled implicitly. On the other hand, the kernel extrapolation is based on matrix approximation under semi-definite constraint, which minimizes the difference of the Von-Neumann Schatten p-norms; while the surrogate kernel uses empirical estimation of the kernel eigenfunctions, and is designed explicitly to preserve the eigen-structure of the given kernel matrix.

4.2.1 Kernel Matrix Alignment Via Surrogate Kernels

The notion of surrogate kernel allows us to "project" a given kernel matrix defined on some sample set onto an arbitrary new set of samples while preserving the eigen-structures. This then serves as a bridge that allows us to freely compare (henceforth transform among) different kernel matrices, which otherwise would be difficult. In the

following, we propose a parametric transform to rectify the kernel matrix from one domain such that it becomes more aligned to the kernel matrix in another domain, by using surrogate kernel.

suppose $\mathcal{X}_1$ and $\mathcal{X}_2$ come from the training and test data, respectively, with corresponding kernel matrices $K_1 \in \mathbb{R}^{n_1 \times n_1}$ and $K_2 \in \mathbb{R}^{n_2 \times n_2}$ defined by some kernel k. Here we apply a symmetric transformation matrix $T \in \mathbb{R}^{n_1 \times n_1}$ on both sides of the kernel matrix K_1, as

$$\tilde{K}_1 = T' K_1 T, \tag{4.6}$$

with the hope that the transformed kernel matrix is more similar to that in the target domain. Here, the transform (4.6) implicitly enforces a nonlinear transform μ on $\mathcal{X}_1$, i.e., $\tilde{\mathcal{X}}_1 = \mu(\mathcal{X}_1)$, such that by using the kernel trick,

$$\langle \Psi(\tilde{\mathcal{X}}_1), \Psi(\tilde{\mathcal{X}}_1) \rangle = \langle \Psi(\mathcal{X}_1)T, \Psi(\mathcal{X}_1)T \rangle, \tag{4.7}$$

where $\Psi(\mathcal{Z}) = [\psi(z_1), ..., \psi(z_n)]$ for input data $\mathcal{Z} = \{z_1, ..., z_n\}$. From (4.7), we can see that the transform μ underlying (4.6) is actually a linear transformation $\Psi(\tilde{\mathcal{X}}_1) = \Psi(\mathcal{X}_1)T$ in the feature space.

Our task is to make the transformed kernel matrix $\tilde{K}_1$ (4.6) approach K_2, the kernel matrix defined on X_2. Since $\tilde{K}_1$ and K_2 are of different sizes and can not be compared directly, we will first compute the surrogate kernel of K_2 on $\mathcal{X}_1$, i.e., $K^{1 \leftarrow 2} = K_{12} K_2^{-1} K_{21}$. Then, we try to minimize the Frobenius norm of the difference between $\tilde{K}_1$ and $K^{1 \leftarrow 2}$ as follows

$$\min_{T \in \mathbb{R}^{n_1 \times n_1}} \left\| T' K_1 T - K^{1 \leftarrow 2} \right\|_F^2 + \gamma \|T\|_F^2. \tag{4.8}$$

Here the second term controls the complexity of the transformation T, and γ controls the balance between enforcing an exact fit and the model complexity. In the empirical evaltions we will further study the algorithm performance w.r.t. the choice of this parameter. By finding the vanishing derivative of (4.8) w.r.t. T, we have

$$T = K_1^{-\frac{1}{2}} (K^{1 \leftarrow 2} - \frac{1}{2} \gamma K_1^{-1})^{\frac{1}{2}}. \tag{4.9}$$

Note that $A^{\frac{1}{2}}$ for a symmetric and positive semi-definite matrix A can be computed as $A^{\frac{1}{2}} = P D^{\frac{1}{2}} P'$, where P's columns are eigenvectors of A, and D is diagonal with

eigenvalues. Similarly, $A^{-\frac{1}{2}} = PD^{-\frac{1}{2}}P'$. Once the transformation T is obtained, $\tilde{\mathcal{X}}_1$ will be also be determined. By virtue of the kernel trick again, we will be able to utilize $\tilde{\mathcal{X}}_1$ (implicitly) in any kernel-based learning algorithm, as long as we can compute the similarity between $\tilde{\mathcal{X}}_1$ and $\mathcal{X}_2$. This will be discussed in the next subsection.

4.2.2 Cross Domain Similarity

In order to use transformed training data $\tilde{\mathcal{X}}_1$ in kernel-based learning algorithms, we will need to compute the composite kernel matrix defined on a $\tilde{\mathcal{X}}_1 \cup \mathcal{X}_2$:

$$G = \begin{bmatrix} \langle \Psi(\tilde{\mathcal{X}}_1), \Psi(\tilde{\mathcal{X}}_1) \rangle & \langle \Psi(\tilde{\mathcal{X}}_1), \Psi(\mathcal{X}_2) \rangle \\ \langle \Psi(\mathcal{X}_2), \Psi(\tilde{\mathcal{X}}_1) \rangle & \langle \Psi(\mathcal{X}_2), \Psi(\mathcal{X}_2) \rangle \end{bmatrix}$$

By design, we have $\langle \Psi(\tilde{\mathcal{X}}_1), \Psi(\tilde{\mathcal{X}}_1) \rangle = T'KT$, and $\langle \Psi(\mathcal{X}_2), \Psi(\mathcal{X}_2) \rangle = K_2$. In the remaining, we need to compute $\tilde{K}_{12}$, the inner product between transformed training data $\Psi(\tilde{\mathcal{X}}_1)$ and the original test data $\Psi(\mathcal{X}_2)$, both of which lie in an infinite-dimensional feature space. By using (4.7), we have $\Psi(\tilde{\mathcal{X}}_1) = \Psi(\mathcal{X}_1)T$. So $\langle \Psi(\tilde{\mathcal{X}}_1), \Psi(\mathcal{X}_2) \rangle = T'\Psi(\mathcal{X}_1)'\Psi(\mathcal{X}_2) = T'K_{12}$. Therefore, we have the following composite kernel which can be used in any kernel based learning algorithms,

$$G = \begin{bmatrix} T'K_1T & T'K_{12} \\ K_{21}T & K_2 \end{bmatrix}. \tag{4.10}$$

It's easy to verify that the composite kernel is always psd. The complexity of our approach is $O(|\mathcal{X}_1 \cup \mathcal{X}_2|^3)$, which can be further reduced by low rank approximation techniques.

4.2.3 Predictions

After computing the transformed Gram matrix, we can apply it to various kernel methods. We choose as example the kernel ridge regression and support vector machine classification for prediction. For kernel ridge regression, we predict labels for the test data $\mathcal{X}_2$ as $\mathbf{y}_2 = \tilde{K}_{21}(\tilde{K}_1 + \eta\mathbf{I})^{-1}\mathbf{y}_1$. For SVM classification, after using the $(\tilde{K}_1, \mathbf{y}_1)$ to train a standard SVM classifier with Lagrangian multipliers α and the bias term b, we predict labels for the test data by $\mathbf{y}_2 = \tilde{K}_{21}(\alpha \odot \mathbf{y}_1) + b$.

4.3 Experiments

In this section, we examine the performance of our approach through five real world-data sets, including two WiFi-localization data sets (for regression task) and three text classification data sets (for classification task). In all our experiments, we randomly choose 60% samples from one domain[1] for training and also 60% samples from another domain for testing. We repeat the experiments for 10 times and report the average results and their standard deviations.

4.3.1 WiFi Time Data

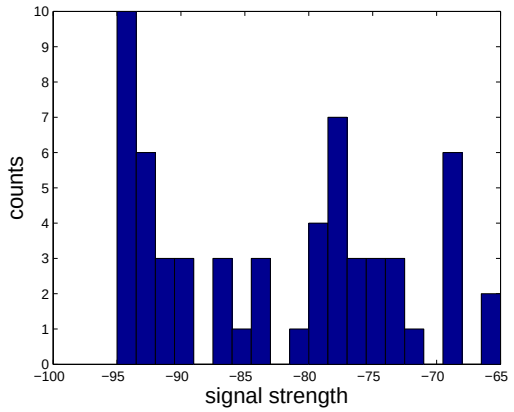
In WiFi-localization, given a set of WiFi signals (samples) $\mathcal{X}=\{x_i \in \mathbb{R}^d\}$ and corresponding locations/labels $\mathcal{Y}=\{y_i \in \mathbb{R}\}$[2], we try to learn a mapping function from the signal space to the location space. However, in practice the WiFi signal data $\mathcal{X}_1$ collected some time before may be out-of-date and will follow a different distribution with the up-to-date WiFi signal data $\mathcal{X}_2$. Figure 4.2 demonstrates the WiFi signal over different time periods, showing the need for distribution matching.

Our experimental setting is as follows. Given WiFi data from two different time periods in the same hallway. In the first time period the data are labeled training data $(\mathcal{X}_1, \mathcal{Y}_1)$, and the second time period are test data $\mathcal{X}_2$. Our goal is to predict the labels for $\mathcal{X}_2$. We compare our approach with 3 others methods: (1) Kernel Ridge Regression (KRR). (2) Transductive Laplacian Regularized Least Square (LapRLS) [8]. (3)Penalized LMS Regression by Kernel Mean Matching (KMMR) [34]. (4)KLIEP[80] (5)TCA[59] (6)GFK[29] (7)SSA[26]

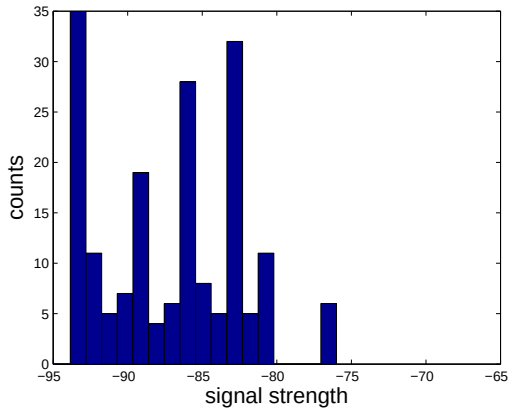
Following the customs in wireless sensor network community, we transform the regression errors to localization accuracy as follows. For each signal to be localized, the accuracy is 100% if the predicted position is within a threshold from the true position, and 0% otherwise. We set this threshold as 3 meters, and report the average accuracy among all the signals. We conduct experiments using 3 time periods, denoted by t1,

[1] e.g. one time period in Section 4.3.1, one wireless device in 4.3.2, or one text data category in 4.3.4.

[2] For simplicity, here we consider localization in some hallway, whose location coordinate is 1-D.



(a) WiFi signal at time period 1



(b) WiFi signal at time period 2

Figure 4.2: WiFi signal variations over time at a fixed location.

t2 and t3. Results (the mean and standard deviation of the accuracy) are reported in Table 4.2. We also conduct paired t-test on the difference between the accuracies of our approach and the other 3 methods. For all three transfer-learning tasks in different time periods, our approach is statistically better than others with the confidence level of at least 99.99%.

4.3.2 WiFi Device Data

The experiments on WiFi Device Data aim to conduct localization on different devices. Note that different wireless devices usually have different signal sensing capacities, and consequently the signals from different devices will vary from each other. Figure 4.3

Table 4.1: WiFi Time data

	$\lvert D_{Scr}\rvert$	$\lvert D_{tar}\rvert$	feature#
t1-vs-t2	792	792	67
t1-vs-t3	792	792	67
t2-vs-t3	792	792	67

Table 4.2: Localization using WiFi Time data (regression).

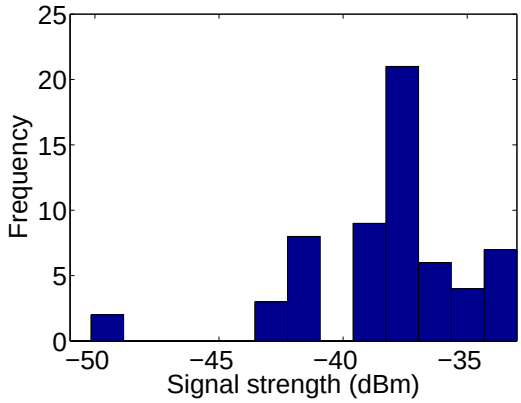
	t1-vs-t2	t1-vs-t3	t2-vs-t3
KRR	80.84±1.14	76.44±2.66	67.12±1.28
LapRLS	82.35±1.08	**94.96±1.04**	85.34±1.88
KMMR	81.84±1.25	76.42±2.64	69.24±1.67
KLIEP	82.67±1.32	75.54±1.15	70.21±1.05
TCA	86.85±1.61	80.48±2.73	72.02±1.32
GFK	89.33±1.27	90.82±1.98	**87.54±1.45**
SSA	87.33±1.29	84.82±1.34	77.54±1.39
Ours	**90.36±1.22**	**94.97±1.29**	**85.83±1.31**

illustrates this by showing the signals collected from two different devices at the same location. This again necessitates the use of data distribution matching over different devices.

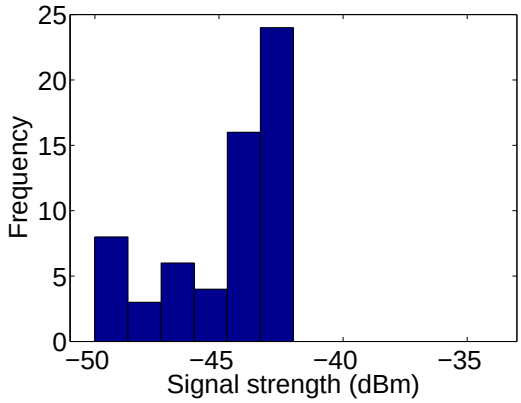
In our experiments, we collect WiFi data by two different devices for 3 hallways indexed from 1 to 3. The first device creates labeled training data $(\mathcal{X}_1, \mathcal{Y}_1)$, and the second device creates test data $\mathcal{X}_2$. For each hallway, we perform distribution matching between the two devices, and the goal is to predict the locations for $\mathcal{X}_2$. We use similar settings as in Section 4.3.1. The threshold error-distance is set as 6 meters. Results are reported in Table 4.4. Again, our approach demonstrates better performance in terms of the localization accuracy, with the confidence level that is at least 99.99%.

4.3.3 Comparison with Domain Adaptation Methods

In this section, we compare our method with two popular methods in domain adaptation, namely the Geodesic Flow Kernel (GFK)[29], and Subspace Alignment (SSA)[26].



(a) WiFi signal at device 1



(b) WiFi signal at device 2

Figure 4.3: WiFi signal variations over devices at a fixed location.

The GFK models the domain shift from the source domain to the target domain by some intermediate phantom domains. The source and the target domains are represented by two subspaces, respectively. The authors developed an efficient kernel method to harvest an infinite number of intermediate phantom domains. The SSA method is a popular low rank method for domain adaptation, which finds a mapping matrix between two domains by aligning their respective subspaces obtained via PCA. The methods are reported in Table 4.2 and Table 4.4.

As can been seen, for the time-shift experiment, the two domain adaptation methods can outperform KRR, and other transfer learning methods such as KMM, KLIEP, and TCA, which demonstrates the effectiveness of the domain adaptation methods in WiFi localization task. However, our methods still outperforms the domain adaptation

Table 4.3: WiFi Device data

	$\|D_{Scr}\|$	$\|D_{tar}\|$	feature#
Hallway1	750	750	46
Hallway2	917	917	46
Hallway3	792	792	46

Table 4.4: Localization using WiFi Device data (regression).

	Hallway1	Hallway2	Hallway3
KRR	60.02±2.60	49.38±2.30	48.42±1.32
LapRLS	53.68±0.45	56.18±0.59	51.53±1.04
KMMR	55.97±0.80	42.25±1.16	47.36±0.19
KLIEP	48.57±6.77	41.71±4.09	44.84±3.44
TCA	65.93±0.86	62.44±1.25	59.18±0.56
GFK	69.23±2.17	**64.42±4.68**	62.24±3.24
SSA	67.33±1.82	63.82±1.31	57.54±1.27
Ours	**76.36±2.44**	**64.69±0.77**	**65.73±1.57**

methods. In the experiment with WiFi time data, our method achieves highest accuracy in 2 out of 3 tasks. Similarly, in the device-shift experiment, it can be seen that the domain adaptation methods achieves higher accuracy comparing to other state of the art transfer learning methods, but again our method performs the best among all 3 tasks.

4.3.4 Text Data

In this experiment, we use the processed Reuters-21578 corpus[3], a public benchmark data on transfer learning with covariate shift assumption. The original Reuters-21578 data set contains 5 top categories. Among these categories, `orgs`, `people` and `places` are the largest ones. So they are used to construct the data set by using the corpus hierarchy. Specifically, there are three data sets `people-vs-places`, `orgs-vs-places` and

[3]http://apex.sjtu.edu.cn/apex_wiki/dwyak

`orgs-vs-people`. In each data set, each category (e.g. `orgs`) has two sub-categories, one for training and the other for testing. We denote the training data as $(\mathcal{X}_1, \mathcal{Y}_1)$, and the testing data as $\mathcal{X}_2$. The feature values for data $\mathcal{X}$ are measured in terms frequency (TF). Since the documents in two sub-categories can have quite different terms, the partitioned training and testing data will have different distributions and therefore a distribution matching solution is preferred. In the experiments we compare our approach with 3 approaches: (1) Standard Support Vector Machine (SVM). In SVM, we use $(\mathcal{X}_1, \mathcal{Y}_1)$ for training, and directly apply the learned model for testing on $\mathcal{X}_2$. (2) Transductive SVM (TSVM). We use the SVM-Light package[4]. The $\mathcal{X}_2$ is chosen as unlabeled data and combined with $(\mathcal{X}_1, \mathcal{Y}_1)$ for transductive learning. (3) Penalized SVM by Kernel Mean Matching (KMMC) [34]. In KMMC, the data reweighting coefficients are applied to the SVM as shown in Eq.6 of [34].

The classification accuracies are reported in Table 4.5. Statistically, the difference between our approach and others are significant for the `people-vs-places` (at least 95.0% confidence level) and `orgs-vs-people` (at least 99.0% confidence level). For `orgs-places`, our approach is still statistically better than the standard SVM and the KMMC, but remains similar with the transductive SVM. This can be explained by the large amount of non-overlapping features between the `orgs` and the `places`. Note that transductive SVM will utilize the test data in the training phase, while we only use the transformed training data $\tilde{\mathcal{X}}_1$ for training. As can be expected, by using a transductive setting we can further improve our performance. Preliminary results have verified this but are not reported due to space limit.

4.3.5 Impact of the Model Parameters

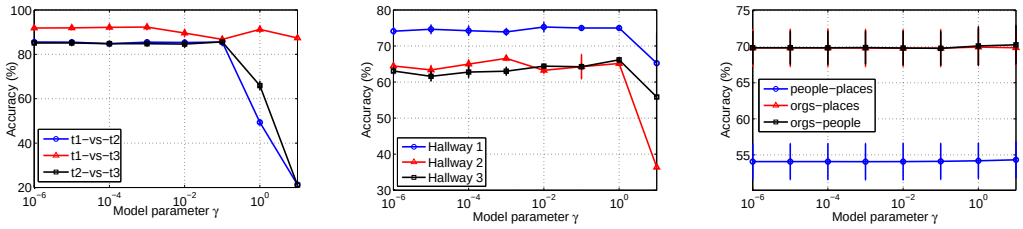
We study the impact of the model parameter γ in Eq. (4.8) over the 3 datasets, as shown in Figure 4.4. For the regression tasks on "WiFi Time" and "WiFi Device", our method favors relatively small γ: when $\gamma \leq 0.1$, the performance is very stable and satisfactory. Note that large values of γ may lead to degraded performance, because when γ is too

[4]http://svmlight.joachims.org/

Table 4.5: Classification on Reuters-21578 corpus.

Data				Accuracy (%)			
index	$\|D_{Scr}\|$	$\|D_{tar}\|$	feature#	SVM	TSVM	KMMC	ours
people-vs-places	1079	1080	8800	53.02±2.64	54.27±2.45	53.23±2.62	**55.65±2.53**
orgs-vs-places	1016	1046	8568	66.83±1.38	64.95±1.82	**68.42±1.38**	**68.58±1.23**
orgs-vs-people	1239	1210	9729	63.61±1.84	63.93±2.60	61.14±2.14	**69.53±2.07**

large, the regularization term in Equation (4.8) may take over the minimization and reduce the contribution of the first term that plays the primary role of distribution matching. For the classification task on "Text", the model performance is shown to be insensitive to γ's change in Figure 4.4(c). Other parameters are common in general kernel methods, so we simply fix them using empirical choices: for the kernel width parameter in the Gaussian kernel, we simply choose it as the average of pairwise squared distances; the regularization parameter C is SVM is fixed at 0.5; the parameter η is chosen as 0.01 in the kernel ridge regression.



(a) WiFi Time (b) WiFi Device (c) Text

Figure 4.4: Impact of model parameter γ over datasets.

Chapter 5

Conclusion

In this thesis, we discussed novel machine learning methods in wireless localization. First we presented a new algorithm for semi-supervised kernel design. Unlike traditional methods that use kernel eigenvectors to span the base kernel and focus on tuning their weights, this work aims at designing high-quality base kernels. In particular, we compute the label-aware eigenvectors via extending the ideal kernel eigenfunction. While eigenvectors from the empirical kernel matrix are computed irrespective of class labels and may be poorly aligned to the target due to various practical factors, computing them based on extrapolating the ideal kernel eigenfunction is more reliable and empirically lead to base kernels of higher quality. The experimental results on real world data sets demonstrated the superiority of our algorithm. An important direction for our future research is to theoretically study the alignment of the label-aware base kernels. In addition, we would explore different ways for propagating the ideal kernel and combining multiple kernels from multiple sources.

Secondly, we presented a distribution matching scheme in the kernel-induced feature space to solve the problem of cross domain indoor localization, with specific emphasis and consideration on kernel based learning methods. We introduced a useful concept called surrogate kernel based on the fundamental theorem underlying the RKHS, such that different kernel matrices can be compared and manipulated directly to realize the feature space distribution matching. Our method demonstrated satisfactory performance on real world data sets. With the surrogate kernels bridging the gap between different Gram matrices, in the future we will examine different transforms including both parametric and non-parametric ones. We will also study the choice the reflectance

data on which the surrogate kernel matrix is evaluated, and apply the potentially abundant and diverse learning approaches related to kernel methods in our scheme.

References

[1] Aboodi, A., and Tat-Chee, W. Evaluation of wifi-based indoor (wbi) positioning algorithm. In *In MUSIC* (2012).

[2] Al-Ahmadi, A. S., Omer, A. I., Kamarudin, M. R., and Rahman, T. A. Multi-floor indoor positioning system using bayesian graphical models. *Progress In Electromagnetics Research B 25* (2010).

[3] ans M. Korenberg, M. M. A., and Noureldin, A. A consistent zero-configuration gps-like indoor positioning system based on signal strength in ieee 802.11 networks. In *In PLANS* (2012).

[4] Bahl, P., and Padmanabhan, V. Radar: An in-building rf-based user location and tracking system. In *In Proceedings of the Conference on Computer Communications* (2000).

[5] Baker, C. *The Numerical Treatment of Integral Equations*. Clarendon Press, 1977.

[6] Baktashmotlagh, M., Harandi, M., Lovell, B., and Salzmann, M. Unsupervised domain adaptation by domain invariant projection. In *In ICCV* (2013).

[7] Baktashmotlagh, M., Harandi, M., Lovell, B., and Salzmann, M. Domain adaptation on the statistical manifold. In *In CVPR* (2014).

[8] Belkin, M., Niyogi, P., and Sindhwani, V. On manifold regularization. In *Proceedings of the 10th International Workshop on Artificial Intelligence and Statistices* (2005), pp. 17–24.

[9] Belkin, M., Niyogi, P., and Sindhwani, V. Manifold regularization: A geometric framework for learning from labeled and unlabeled examples. *Journal of Machine Learning Research 7* (2006).

[10] Bengio, Y., Delalleau, O., Roux, N. L., Paiement, J.-F., Vincent, P., and Ouimet, M. Learning eigenfunctions links spectral embedding and kernel pca. *Neural Computation 16* (2004).

[11] Caruana, R. Multitask learning. *Machine Learning 28*, 1 (1997), 41–75.

[12] Chan, S., and Sohn, G. Indoor localization using wi-fi based fingerprinting and trilateration techiques for lbs applications. In *In International Conference on 3D Geoinformation* (2012).

[13] Chapelle, O., Weston, J., and Sch?lkopf, B. Cluster kernels for semi-supervised learning. In *Advances in Neural Information Processing Systems* (2003).

[14] Chapelle, O., and Zien, A. Semi-supervised classification by low density separation. In *Proceedings of the 10th International Conference on Artificial Intelligence and Statistics* (2005).

[15] Collobert, R., Sinz, F., Weston, J., Bottou, L., and Joachims, T. Large scale transductive svms. *Journal of Machine Learning Research 7* (2006).

[16] Cortes, C., Kloft, M., and Mohri, M. Learning kernels using local rademacher complexity. In *Advances in Neural Information Processing Systems* (2013).

[17] Cortes, C., Mohri, M., and Rostamizadeh, A. Two-stage learning kernel algorithms. In *Proceedings of the 27th Annual International Conference on Machine Learning* (2010).

[18] Cox, T., and Cox, M. *Multidimensional Scaling*. Chapman Hall, 1994.

[19] Cristianini, N., Kandola, J., Elisseeff, A., and Shawe-Taylor, J. On kernel-target alignment. In *Advances in Neural Information Processing Systems* (2002).

[20] de Silva, V., and Tenenbaum, J. Global versus local methods in nonlinear dimensionality reduction. In *In Advances in Neural Information Processing Systems 15* (2002).

[21] Donahue, J., Jia, Y., Vinyals, O., Hoffman, J., Zhang, N., Tzeng, E., and Darrell, T. Decaf: A deep convolutional activation feature for generic visual recognition. In *In ICML* (2014).

[22] DoWoo, P., and Goo, P. J. An enhanced ranging scheme using wifi rssi measurements for ubiquitous location. In *In CNSI* (2011).

[23] Drineas, P., and Mahoney, M. On the nystrom method for approximating a gram matrix for improved kernel-based learning. *Journal of Machine Learning Research 6* (2005).

[24] Faloutsos, C., and Lin, K. Fastmap: A fast algorithm for indexing, data mining and visualization of traditional and multimedia datasets. In *In Proceedings of the 1st ACM SIGMOD International Conference on Management of Data* (1995).

[25] Fang, S., and Lin, T. Principal component localization in indoor wlan environments. *IEEE Transactions on Mobile Computing 11* (2012).

[26] Fernando, B., Habrard, A., Sebban, M., and Tuytelaars, T. Unsupervised visual domain adaptation using subspace alignment. In *In CVPR* (2013).

[27] Ferris, B., Hahnel, D., and Fox, D. Adaptive localization in a dynamic wifi environment through multi-view learning. In *In Proceedings of the 22nd AAAI Conference on Artificial Intelligence* (2007).

[28] Fowlkes, C., Belongie, S., Chung, F., and Malik, J. Spectral grouping using the nystrom method. *IEEE Transactions on Pattern Analysis and Machine Intelligence 26* (2004).

[29] Gong, B., Shi, Y., Sha, F., and Grauman, K. Geodesic flow kernel for unsupervised domain adaptation. In *In CVPR* (2012).

[30] Gopalan, R., Li, R., and Chellappa, R. Domain adaptation for object recognition: An unsupervised approach. In *In ICCV* (2011).

[31] Han, D., Jung, S., Lee, M., and Yoon, G. Building a practical wi-fibased indoor navigation system. *IEEE Pervasive Computing 13* (2014).

[32] He, X., Ji, M., Zhang, C., and Bao, H. A variance minimization criterion to feature selection using laplacian regularization. *IEEE Transactions on Pattern Analysis and Machine Intelligence 33* (2011).

[33] Huang, D., and Wang, Y. F. Coupled dictionary and feature space learning with applications to cross-domain image synthesis and recognition. In *In ICCV* (2013).

[34] Huang, J., Smola, A., Gretton, A., Borgwardt, K., and Scholkopf, B. Correcting sample selection bias by unlabeled data. In *Advances in Neural Information Processing Systems 19* (2007), pp. 601–608.

[35] III, H. D., and Marcu, D. Domain adaptation for statistical classifiers. *Journal of Artifical Intelligence Research 26* (2006), 101–126.

[36] Joachims, T. Transductive inference for text classification using support vector machines. In *Proceedings of the 16th Annual International Conference on Machine Learning* (1999).

[37] Khodayari, S., Maleki, M., and Hamedi, E. A rss-based fingerprinting method for positioning based on historical data. In *In SPECTS* (2012).

[38] Kondor, R. I., and Lafferty, J. D. Diffusion kernels on graphs and other discrete input spaces. In *Proceedings of the 19th Annual International Conference on Machine Learning* (2002).

[39] Kulis, B., Basu, S., Dhillon, I., and Mooney, R. Semi-supervised graph clustering: a kernel approach. In *Proceedings of the 22th Annual International Conference on Machine Learning* (2005).

[40] Kulis, B., Saenko, K., and Darrell, T. What you saw is not what you get: Domain adaptation using asymmetric kernel transforms. In *In CVPR* (2011).

[41] Kwok, J. T., and Tsang, I. W. Learning with idealized kernels. In *Proceedings of the 20th Annual International Conference on Machine Learning* (2003).

[42] Lanckriet, G. R. G., Cristianini, N., Bartlett, P., Ghaoui, L. E., and Jordan, M. I. Learning the kernel matrix with semidefinite programming. *Journal of Machine Learning Research 5* (2004).

[43] Lawrence, M., Seeger, N., and Herbrich, R. Fast sparse gaussian process methods: the informative vector machine. In *In Advances in Neural Information Processing Systems 16* (2003).

[44] Letchner, J., Fox, D., and LaMarca, A. Large-scale localization from wireless signal strength. In *In Proceedings of the 20th National Conference on Artificial Intelligence* (2005).

[45] Li, Y., and Zhou, Z. Towards making unlabeled data never hurt. In *Proceedings of the 28th Annual International Conference on Machine Learning* (2011).

[46] Li, Z., Liu, J., and Tang, X. Pairwise constraint propagation by semidefinite programming for semi-supervised classification. In *Proceedings of the 25th Annual International Conference on Machine Learning* (2008).

[47] Liu, H., Gan, Y., Yang, J., Sidhom, S., Y. Wang, Y. C., and Ye, F. Zee: Zero-effort crowdsourcing for indoor localization. In *In ACM MobiCom* (2012).

[48] Liu, H., Yang, J., Sidhom, S., Wang, Y., Chen, Y., and Ye, F. Accurate wifi based localization for smartphones using peer assistance. *IEEE Transactions on Mobile Computing 13* (2014).

[49] Long, M., Ding, G., Wang, J., Sun, J., Guo, Y., and Yu, P. Transfer sparse coding for robust image representation. In *In CVPR* (2013).

[50] Maligan, D., Elnahrawy, E., Martin, R., Ju, W., Krishnan, P., and Krishnakumar, A. Bayesian indoor positioning systems. In *In Proceedings of the Conference on Computer Communications* (2005).

[51] Melacci, S., and Belkin, M. Laplacian support vector machines trained in the primal. *Journal of Machine Learning Research 12* (2011).

[52] Ng, A. Y., Jordan, M. I., and Weiss, Y. On spectral clustering: Analysis and an algorithm. In *Advances in Neural Information Processing Systems* (2001).

[53] Nguyen, X., Jordan, M. I., and Sinopoli, B. A kernel-based learning approach to ad hoc sensor network localization. *ACM Transactions on Sensor Networks 1* (2005).

[54] Ni, J., Qiu, Q., and Chellappa, R. Subspace interpolation via dictionary learning for unsupervised domain adaptation. In *In CVPR* (2013).

[55] Oquab, M., Laptev, I., Bottou, L., and Sivic, J. Learning and transferring midlevel image representations using convolutional neural networks. In *In CVPR* (2014).

[56] Pan, J. J. Learning-based localization in wireless and sensor networks. In *PhD Thesis* (2007).

[57] Pan, S. J. Feature-based transfer learning with real-world applications. In *PhD Thesis* (2010).

[58] Pan, S. J., Kwok, J. T., Yang, Q., and Pan, J. Gaussian processes for signal strength-based location estimation. In *In Proceedings of Robotics: Science and Systems* (2006).

[59] Pan, S. J., Tsang, I., Kwok, J., and Yang, Q. Domain adaptation via transfer component analysis. *IEEE Transactions on Neural Networks 22* (2007), 199–210.

[60] Pereira, C., Guenda, L., and Carvalho, B. N. A smart-phone indoor/outdoor localization system. In *In IPIN* (2011).

[61] Platt, J. Fastmap, metricmap, and landmark mds are all nystrom algorithms. In *In Proceedings of the 10th International Workshop on Artificial Intelligence and Statistics* (2005).

[62] Rai, A., Chintalapudi, K. K., Padmanabhan, V. N., and Sen, R. Zee: Zero-effort crowdsourcing for indoor localization. In *In ACM MobiCom* (2012).

[63] R.Hansen, R.Wind, Jensen, C. S., and B.Thomsen. Algorithmic strategies for adapting to environmental changes in 802.11 location fingerprinting. In *In IPIN* (2010).

[64] Saenko, K., Kulis, B., Fritz, M., and Darrell, T. Adapting visual category models to new domains. In *In ECCV* (2010).

[65] Savvides, A., Han, C., and B.Strivastava, M. Dynamic fine-grained localization in ad-hoc networks of sensors. In *In Proceedings of the 7th Annual International Conference on Mobile Computing and Networking* (2001).

[66] Scholkopf, B., and Smola, A. J. *Learning with Kernels Support Vector Machines, Regularization, Optimization, and Beyond.* MIT Press, 2001.

[67] Shawe-Taylor, J., and Cristianini, N. *Kernel Methods for Pattern Analysis.* Cambridge University Press, 2004.

[68] Shawe-taylor, J., Holloway, R., and Williams, C. K. I. The stability of kernel principal components analysis and its relation to the process eigenspectrum. In *Advances in Neural Information Processing Systems* (2003).

[69] Shawe-Taylor, J., Williams, C., Cristianini, N., and Kandola, J. On the eigenspectrum of the gram matrix and the generalization error of kernel-pca. *IEEE Transactions on Information Theory 51* (2005), 2510–2522.

[70] Shekhar, S., Patel, V., Nguyen, H. V., and Chellappa, R. Generalized domain adaptive dictionaries. In *In CVPR* (2013).

[71] Shen, G., Chen, Z., Zhang, P., Moscibroda, T., and Zhang, Y. Walkie-markie: indoor pathway mapping made easy. In *In USENIX NSDI* (2013).

[72] Shen, G., Chen, Z., Zhang, P., Moscibroda, T., and Zhang, Y. Walkie-markie: indoor pathway mapping made easy. In *In USENIX NSDI* (2013).

[73] Shi, J., and Malik, J. Normalized cuts and image segmentation. *IEEE Trasactions on Pattern Analysis and Machine Intelligence 22* (2000).

[74] Shimodaira, H. Improving predictive inference under covariate shift by weighting the log-likelihood function. *Journal of Statistical Planning and Inference 90* (2000), 227–244.

[75] Shrivastava, A., Shekhar, S., and Patel, V. Unsupervised domain adaptation using parallel transport on grassmann manifold. In *In IEEE Winter conference on Applications of Computer Vision* (2014).

[76] Sinha, K., and Belkin, M. Semi-supervised learning using sparse eigenfunction bases. In *Advances in Neural Information Processing Systems* (2009).

[77] Smola, A. J., and Kondor, R. Kernels and regularization on graphs. In *In Conference on Learning Theory* (2003).

[78] Smola, A. J., and Scholkopf, B. A tutorial on support vector regression. *Journal Statistics and Computing 14* (2004).

[79] Sugiyama, M., Meguro-ku, Krauledat, M., and Muller, K. Covariate shift adaptation by importanceweighted cross validation. *Journal of Machine Learning Research* (2007), 985–1005.

[80] Sugiyama, M., Nakajima, S., Kashima, H., Bunau, P., and Kawanabe, M. Direct importance estimation with model selection and its application to covariate shift adaptation. In *Advances in Neural Information Processing Systems 20* (2008), pp. 1433–1440.

[81] Sun, W., Liu, J., Wu, C., Yang, Z., Zhang, X., and Liu, Y. Moloc: On distinguishing fingerprint twins. In *In IEEE ICDCS* (2013).

[82] Thrun, S., Fox, D., Burgard, W., and Dellaert, F. Robust monte carlo localization for mobile robots. *Artificial Intelligence 128* (2001).

[83] Thrun, S., and Mitchell, T. Learning one more thing. In *Proceedings of the 4th International Joint Conference on Artificial Intelligence* (1995).

[84] Vishwanathan, S., Borgwardt, K., Guttman, O., and Smola, A. Kernel extrapolation. *Nerocomputing 69* (2006), 721–729.

[85] Wang, H., Sen, S., Elgohary, A., Farid, M., Youssef, M., and Choudhury, R. R. No need to war-drive: Unsupervised indoor localization. In *In ACM MobiSys* (2012).

[86] Wang, L., Wang, X., Lin, K., Shasha, D., Shapiro, B. A., and Zhang, K. Evaluating a class of distance-mapping algorithms for data mining and clustering. In *In Proceedings of 5th ACM SIGKDD international conference on Knowledge discovery and data mining* (1999).

[87] Wang, S., Zhang, L., Liang, Y., and Pan, Q. Semi-coupled dictionary learning with applications in image super-resolution and photo-sketch synthesis. In *In CVPR* (2012).

[88] Williams, C., and Seeger, M. The effect of the input density distribution on kernel-based classifiers. In *Proceedings of the 17th International Conference on Machine Learning (ICML 2000)* (2000), pp. 1159–1166.

[89] Williams, C., and Seeger, M. Using the nystrom method to speed up kernel machines. In *Advances in Neural Information Processing Systems* (2001).

[90] Wu, C., Yang, Z., Xiao, C., Yang, C., Liu, Y., and Liu, M. Static power of mobile devices: Self-updating radio maps for wireless indoor localization. In *In IEEE INFOCOM* (2015).

[91] Yang, Q., Pan, S. J., and Zheng, V. W. Estimating location using wi-fi. *IEEE Intelligent Systems 23* (2008).

[92] Yang, Z., Wu, C., and Liu, Y. Locating in fingerprint space: Wireless indoor localization with little human intervention. In *In ACM MobiCom* (2012).

[93] Yin, J., Yang, Q., and Ni, L. Learning adaptive temporal radio maps for signalstrength-based location estimation. *IEEE Transactions on Mobile Computing 7* (2008).

[94] Z. Farid, R. N., and Ismail, M. Recent advances in wireless indoor localization techniques and system. *Journal of Computer Networks and Communications 2013* (2013).

[95] Z. Yang, C. W., and Liu, Y. Locating in fingerprint space: wireless indoor localization with little human intervention. In *In Mobicom* (2012).

[96] Zadrozny, B. Learning and evaluating classifiers under sample selection bias. In *Proceedings of the 21th International Conference on Machine Learning (ICML 2004)* (2004), pp. 903–910.

[97] Zheng, J., Liu, M., Chellappa, R., and Phillips, P. J. A grassmann manifold-based domain adaptation approach. In *In ICPR* (2012).

[98] Zhou, X., and Belkin, M. Semi-supervised learning by higher order regularization. In *The 14th International Conference on Artificial Intelligence and Statistics* (2011).

[99] Zhu, X., Ghahramani, Z., and Lafferty, J. Semi-supervised learning using gaussian fields and harmonic functions. In *Proceedings of the 20th Annual International Conference on Machine Learning* (2003).

[100] Zhu, X., Kandola, J., Ghahramani, Z., and Lafferty, J. Nonparametric transforms of graph kernels for semi-supervised learning. In *Advances in Neural Information Processing Systems* (2004).