國立臺灣大學電機資訊學院資訊工程學研究所

博士論文

Department of Computer Science and Information Engineering
College of Electrical Engineering and Computer Science
National Taiwan University

Doctoral Dissertation

知識轉移於智慧家庭環境之行為辨識應用

Building Activity Recognition Models Using Transfer Learning in Smart Home Environment

蔣益庭

Yi-ting Chiang

指導教授：許永真 博士

Advisor: Jane Yung-jen Hsu, Ph.D.

中華民國 102 年 1 月

January, 2013

國立臺灣大學博士學位論文
口試委員會審定書

知識轉移於智慧家庭環境之行為辨識應用

Building Activity Recognition Models Using Transfer Learning in Smart Home Environment

本論文係蔣益庭君（學號 D94922021）在國立臺灣大學資訊工程學系完成之博士學位論文，於民國 102 年 1 月 9 日承下列考試委員審查通過及口試及格，特此證明

口試委員：

（指導教授）

系 主 任

Acknowledgments

I would like first and foremost thank my advisor, Prof. Jane Yung-jen Hsu, for her guidance and support. I also would like to thank Wan-rong for her generosity and consideration. Next, I would like to express my gratitude to Yi-Ting and Prof. Lu for their help on my thesis and oral examination. I would like to thank my colleague Wen-Chieh, who work with me at the BL lab, and all the students in our lab who helped. I'd like to give special thanks to Prof. Da-Wei Wang, Prof. Churn-Jung Liau, and Prof. Tsan-sheng Hsu at Massive Data Computation and Management Laboratory in Institute of Information Science of Academia Sinica, and all the members in this lab, especially to Wan-Hsuan, Hung-Jui, Meng-Tsung, and Yun-Ching for their support. Without them, this work can not be finished.

Finally, I want to thank my parents for their unconditional love and support.

致謝

這篇論文的完成要歸功於許多人的幫助。首先我要感謝指導教授許永真老師這幾年的指導以及鼓勵。其次我要感謝常常關心我近況的婉容學姐，在論文內容以及口試給我不少建議與幫助的怡亭，小陸，和我在博理館一起努力的文杰，以及實驗室為論文成果以及我的口試幫過忙的所有同學們。此外，我要特別感謝中央研究院資訊科學研究所巨量資料計算暨管理實驗室的王大為老師，廖純中老師，以及徐讚昇老師給我的支援，以及同實驗室的同仁們，尤其是宛萱，紘睿，孟宗，以及雲青的幫助。沒有他們，這篇論文不可能完成。

最後，我要感謝多年以來一直默默支持我完成學業的父母親。謝謝他們。

中文摘要

大多數的行為辨識研究應用了機器學習方法。傳統上，機器學習基於一個假設來建立模型：用來建立模型時的環境與應用此模型的環境是相同的。此假設並非總是成立。若是可以先於一實驗室環境收集資料，並利用轉移學習(Transfer learning)來減低收集資料的花費，建立行為辨識模型於一智慧家庭將更為實用。

轉移學習移除了建立及應用模型時環境必須相同的限制，允許學習及測試模型的資料庫可以相異，並成功地應用於許多機器學習問題。此研究介紹一應用於行為辨識的知識轉移架構。具體來說，我們定義一個以特徵為基礎，可以自動計算新的特徵表述來轉移知識的知識轉移架構。我們於下面兩種情境下實際建造行為辨識模型來驗證此架構。情境一假設資料已標記的來源領域資料庫以及關於來源及目標領域資料庫的背景知識皆可知，但目標資料庫並不可得；情境二假設資料已標記的來源以及目標領域資料庫皆可得。實驗證明此知識轉移架構可在兩相異環境中成功地萃取並轉移知識。

關鍵詞：傑森-向農偏差，轉移學習，行為辨識，機器學習，智慧家庭

Abstract

Most activity recognition research makes use of machine learning methods. Traditional machine methods build a model under the assumption that this model will be applied to the same environment where the training dataset is collected, which is sometimes unrealistic in real world. In addition, collecting data sets in every environment where the model is going to be applied is not feasible. Building activity recognition models in a smart home is more practical if we can collect the dataset in a laboratory environment, and use transfer learning to reduce the effort of data collection.

Transfer learning relaxes this constraint, so that the training and the testing dataset can be different. It has considerable success in many machine learning problems. In this work, we propose a knowledge transfer framework for activity recognition. Specifically, we propose a feature-based knowledge transfer framework which can automatically find the new formulation of features to transfer knowledge between two domains. We apply this framework under two different scenarios to train activity recognition models: In the first scenario, a labelled source dataset is available, and we have the background knowledge about the source and target domain, but we do not have any target domain data samples. In the second scenario, we have both labelled the source and target domain dataset. The experimental results show that this framework can successfully extract and transfer knowledge between two different domains.

Keyword: Jenson-Shannon divergence, transfer learning, activity recognition, machine learning, smart home

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Population ageing is a challenge occurring all over the world, thus it is becoming more and more important to tend to the needs of the increasing elderly population. A smart home system which can provide various services is very valuable for elders who live alone. The smart home system would be able to provide daily services, recognize abnormalities, and trigger alarm if necessary. In order to offer appropriate services, activities that residents are performing is a key context. Therefore, activity recognition is vital to many context-aware applications in an intelligent environment.

Most activity recognition research makes use of machine learning methods. In supervised learning, the model is trained from the dataset consisted of features and labels. The scenario in activity recognition making use of supervised learning algorithm is that a model is trained using the dataset collected from the environment, and is applied to recognize what activity is performed. Features are usually extracted from the raw data of the sensors that are deployed in the environment, and the labels are the activities performed. Conventionally, the model is used in the environment which is the same with where it is trained. Therefore, in order to learn a model which will be used in the environment, the dataset collected from this environment is necessary.

However, collecting and labelling data in houses can disturb the daily life of the users, or even change their behaviour. Therefore it is also necessary to take the effect of the sensors on users in the environment into consideration, as discussed in [44]. In addition, data collection and labelling each sample can be very costly. Therefore, it would be more practical if we can reduce the effort of collecting data in another environment before deploying the smart home system for practical use. For example, collecting most of the data in a laboratory environment and only a few necessary data in the house where the users live. However, the layout of the laboratory environment and the deployed sensors may be different from the real environment. In this case, traditional machine learning methods are not applicable.

Even if the layout of the environment and the sensors used to collect data are the same, there may still be problems in applying traditional machine learning methods, because the behaviour of different individuals may differ. Even when performing the same activity, people can display a diverse range of behaviour patterns. The divergence here is on people, not on sensors. Besides, for a smart home environment, people can always add or remove furniture or appliances as they wish. It is impossible to guarantee that the environment is invariant, so it is inevitable for any intelligent system in a smart home to deal with the problem of the difference between the training and the testing datasets.

In transfer learning, the environment of the training and testing datasets can be different. Knowledge is transferred from a source domain to a target domain. Only reusable knowledge is extracted from the source domain to facilitate the task in the target domain. Transfer learning deals with the problem of not only knowledge extraction but also knowledge evaluation to facilitate the learning process. Building the activity recognition model for a smart home environment will benefit from a kind of knowledge transfer mechanism which can extract and transfer knowledge correctly. On the other hand, if we inappropriately transfer knowledge between two domains, the transferred knowledge will not only fail to facilitate the learning process, but also hurt the model performance. This phenomenon is called negative transfer, which should be prevented when applying transfer learning method in any learning tasks.

In this work, we proposed a feature-based knowledge transfer framework to help to build the activity recognition model under a smart home environment. We empirically show that this framework can appropriately reuse the extracted knowledge under different scenarios when we deploy a smart home system into an environment.

1.1 Problem Definition

It is necessary to bring in transfer learning when there are differences between the source and the target domain. In the aspect of smart home environments, the difference between two environments is mainly on

1. Collection of sensors: the number of sensors, the specification of sensors, sensor types, and the layout of the environment.

2. Activity: the set of activities performed in the environment.

3. Resident: the habits and the patterns of behaviour of the residents

Formally, let X_s and X_t be the source and the target domain feature sets, and Y_s and Y_t be the corresponding label sets. The three cases can be represented as

1. Collection of sensors: $X_s \neq X_t$.

2. Activity: $Y_s \neq Y_t$.

3. Resident: $P(X_s) \neq P(X_t)$, $P(Y_s) \neq P(Y_t)$, and $P(Y_s|X_s) \neq P(Y_t|X_t)$.

In this work, we focus on the first case, that is, the difference of the two domains is on the collection of sensors. Therefore, $X_s \neq X_t$. We assume that residents perform the same set of activities in the environment. In addition, it is assumed that there is only one resident in the environment, and we do not take into consideration the divergence of behaviour between the resident in both environments. We only deal with the difference of the set of sensors between two domains.

1.1.1 Scenario Settings

There are two scenarios which we will be concerning:

1. The dataset collected in a laboratory environment is available, and we are preparing to deploy sensors to the target domain environment.

2. The dataset collected in a laboratory environment is available, and we have collected some samples in the target environment.

In the first scenario, we have not started to collect data in the target domain, but we assume that the background knowledge about the sensors in the two environments is available. Namely, we know what kind of sensors will be there in the two environments. We also know the information about the layouts of the environments and where these sensors are going to be deployed. The input under this scenario is a labelled dataset in the source domain, and the background knowledge about both the source and the target domain. In the second scenario, the inputs are two labelled datasets.

The output of our work is a description about how to reformulate the source and the target domain datasets to transfer learnt knowledge. This description can be used to create a new source domain dataset to train activity recognition models, and to convert the samples collected in the target domain.

Using ambient sensors instead of cameras or microphones is generally preferred in a smart home environment due to privacy issues. Moreover, using wearable sensors may affect the applicability of the smart home system. Therefore, the experiments in this work are carried out using the datasets that only contain data samples from ambient sensors.

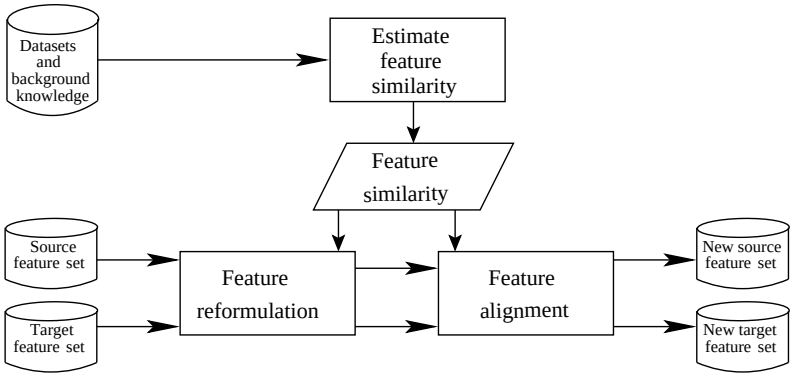


Figure 1.1: The feature-based knowledge framework implemented in this work.

1.2 Proposed Solution

An overview of our proposed framework is shown in figure 1.1 to transfer knowledge to solve the activity recognition problem. The framework is a feature-based framework which is based on the assumption that the feature similarity between the two domains is given (as the domain knowledge) or computable (from the datasets). The feature similarity in figure 1.1 essentially gives the degree of similarity between features, which is vital to all other procedures in the framework.

We use two procedures to compute the description of how to reformulate feature sets. First, in the same dataset, we first combine similar features together and dissimilate these features, which are done by feature reformulation. Second, between two datasets, we compute a mapping of features according to feature similarity in the feature alignment procedure. With these two procedures, we get a new feature set. All the datasets will be represented using the new feature set to transfer knowledge. The detailed description of the framework is given in chapter 3.

The reason for naming this framework as *feature-based* framework is that knowledge evaluation is executed on each features, not on the full feature set. The information extracted from one feature is evaluated and compared with the information extracted from another feature.

We say two features are similar if the provide highly correlated information to infer the label.

1.3 Contribution

The proposed feature-based knowledge transfer framework can effectively extract knowledge from one domain and make use of it in another. This feature-based framework provides a new method which can transfer learnt knowledge as well as give a possible

direction to evaluate the quality of the transferred knowledge. We will show that this framework can be used under two different scenarios—when only the dataset collected in the laboratory environment is available, and when the sensors deployed in the target environment have started to collected some data samples. It is possible to choose how to transfer knowledge according to the situation at hand. Moreover, the feature-based knowledge transfer framework may also be used to give a quantification of the divergence between two datasets. If there are more than one candidate source domain datasets, it is possible to give a measure to select a better one to transfer knowledge.

1.4 Overview

In this work, we will review related studies in activity recognition using transfer learning in chapter 2. Since feature similarity is one of the main component in this framework, we will also review some methods which measure similarity of different kinds of objects. After that, we describe the proposed feature-based knowledge transfer framework in chapter 3. The detailed procedures of this framework are given in section 3.2 and 3.3. The experiments we performed, including the description of datasets, detailed data preprocessing methods, parameter settings, and the experimental results are given in chapter 4. Finally, the advantages and the disadvantages of our proposed framework is discussed, and possible extensions are given in chapter 5.

Chapter 2

Related Work

2.1 Transfer Learning

Traditional machine learning focuses on learning from datasets that are assumed to have the same distribution as the prediction target, namely, the two task are the same. Recently, there has been research on generalizing and utilizing learned knowledge to not only identical tasks but also related ones in machine learning, which has become an important research area which makes machine learning more applicable to applications in the real world.

Multi-task learning, for example, aims to learn several tasks at the same time [7]. Transfer learning focuses on learning information from one environment to facilitate the learning process on another task [46]. Among these studies, most of them are applied to the scenario that the feature sets or the label sets of the datasets are different, but the problem domain or the task is related. For example, datasets of the white wine and red wine are respectively used as the source and target domain dataset to predict the quality of the wine in [6]. A method that can be used to extract knowledge from one handwriting-digit dataset to an English handwriting dataset is proposed in [40]. Research under this kind of scenario can also be found in [36, 53, 54, 55, 29, 38]. On the other hand, there are also some researchers proposed methods to utilize knowledge between different problem domains. In cognitive science, this kind of learning has been studied that knowledge is transferred through domain analogy [19, 16, 27]. In machine learning, researchers have also proposed some methods dealing with this kind of learning in [51, 35, 32, 34]. The theoretical analyses on the probability and limitation of learning between different tasks have been studied in [1, 2, 3, 13, 4]. The categorization of transfer learning can be found in [37]. A survey about transfer learning in different learning models, especially in reinforcement learning, is available in [47].

2.2 Transfer Learning in Activity Recognition

Recently, there are a number of studies applying transfer learning to activity recognition that uses ambient sensors in smart home environment, and a large proportion of which is on the divergence of the source and target domain between what and how sensors are used to collect the datasets. Most of them make the same assumption that similar sensors or features play similar roles in the tasks. Therefore, they find the pairs of sensors or features which are similar to transfer knowledge.

In [49] and [50], knowledge is transferred by identifying the properties of sensors. In [49], function groups of sensors are defined and used to manually map the sensor data. After that, a semi-supervised learning method is used to update parameters in hidden Markov model (HMM) [39]. The semi-supervised learning method used in this research is Expectation Maximization (EM) algorithm which make use of both labelled and unlabelled data samples. In [50], they defined meta features for sensors in datasets. The meta features of sensors are used to decide the mapping of sensors between two datasets. Hyper parameters of HMM parameters are defined, and labelled source domain data samples are used to estimate these hyper parameters. After that, EM algorithm is used to learn the parameters and hyper parameters by using unlabelled and labelled (if available) target domain data samples. The function groups and meta features in these two papers are defined according to prior knowledge of sensors in the environment.

Instead of mapping and merging sensor data according to prior knowledge about the environment, an iterative parameter updating algorithm, HHTL (Home to Home Transfer Learning), is proposed to find the mapping between not only sensors but also activities of different houses in [41]. In that paper, they define mapping matrices for both sensors and activities. These mapping matrices are computed iteratively from source and target domain datasets, not from background knowledge of sensors.

Another scenario of transfer learning in activity recognition is under the assumption that the set of sensors in source and target domains are the same, but activities performed are different. In [22], sensor data in source and the target domains share the same feature space. The difference in the two domains is on their label spaces which are the set of the performed activities. Therefore, it is necessary to find out the relationship between two activities in different domains. In their paper, web mining technique is used to compute the similarity of activities. These estimated similarity and source domain datasets are then used to create a pseudo training dataset. A target domain dataset is not necessary.

In addition to the smart home environment, activity recognition can be applied to other kind of environments, and transfer learning can also be used if necessary. For example, restricted Boltzmann machines (RBMs) is used in [52] to transfer knowledge between

two domains with the same labels but different features. Feature sets of source and target domains are both mapped to a common feature set, which is a hidden layer in RBMs. As a result, similar features are automatically grouped together by the RBMs. Unlike the work we mentioned above, datasets used in this work are not collected from a smart home environment. Information about activities is extracted from some on-line dictionaries to help recognize both known and unknown activities in video clips, which are the target domain dataset. Their method is applied not only to activity recognition but also to a cross-lingual sentiment analysis problem. A survey about transfer learning in activity recognition is available in [11].

2.3 Similarity Measures

In one of our scenarios, it is assumed that we the have background knowledge of sensors in both source and target domains. If similarity of features is explicitly given, transferring knowledge from one domain to another would be easier. But in most cases, information about feature similarity is not available. We introduce some similarity and divergence measures of different kinds of objects in the following section.

2.3.1 For i.i.d. Random Variables

Given two random variables X and Y. If all the samples in X and Y are independent and identically distributed (i.i.d.), a commonly used measures for the relatedness of the two random variables is correlation coefficient. Correlation coefficient evaluates degree of linear dependency of X and Y. Assume we have pairs of data $\{(x_1, y_1), \ldots, (x_n, y_n)\}$ sampled independently from a function $p(X, Y)$. Correlation coefficient of X and Y is

$$\rho_{XY} = \frac{\mathrm{Cov}(X,Y)}{\sqrt{\mathrm{Var}(X)}\sqrt{\mathrm{Var}(Y)}} \tag{2.1}$$

$$= \frac{\sum_k (x_k - \bar{x})(y_k - \bar{y})}{\sqrt{\sum_k (x_k - \bar{x})^2}\sqrt{\sum_k (y_k - \bar{y})^2}} \tag{2.2}$$

where $\mathrm{Var}(X)$ and $\mathrm{Var}(Y)$ are the sample variances, and $\bar{x} = \frac{1}{n}\sum_i x_i$ and $\bar{y} = \frac{1}{n}\sum_i y_i$ are sample means of X and Y respectively. $\mathrm{Cov}(X, Y)$ is covariance of X and Y.

In information theory, we can estimate the amount of information of random variables from their distribution. Mutual information of two random variables X, Y is used to evaluate the amount of common information in them, which is defined as

$$I(X;Y) = \sum_{X,Y} p(X,Y) \log \frac{p(X,Y)}{p(X)p(Y)}$$

If X and Y contain no common information, $I(X;Y) = 0$. In this case, we would say that X and Y are not similar.

On the other hand, if we want to evaluate the divergence of two random variables by their probability function P and Q, we can use KL (Kullback-Leibler) divergence [28]. KL divergence, or relative entropy, of distributions P and Q is defined as

$$D(P||Q) = \sum_x P(x) \log \frac{P(x)}{Q(x)} \tag{2.3}$$

It can be shown that $D(P||Q) \geq 0$ with equality if and only if $P = Q$.

Although KL divergence is a commonly used method in many research areas, it has some disadvantages. First, as we can observe from equation 2.3, KL divergence is not symmetric, which means that $D(P||Q) \neq D(Q||P)$. Moreover, the value of KL divergence is not bounded above. Some new divergence measures have been proposed based on KL divergence. One of these measures is Jenson-Shannon divergence [15], $\mathrm{JSD}(P, Q)$, which is defined as

$$\mathrm{JSD}(P||Q) = \frac{1}{2}(\mathrm{KL}(P||R) + \mathrm{KL}(Q||R)), \tag{2.4}$$

where $R = \frac{P+Q}{2}$.

It is easy to see that Jenson-Shannon divergence is symmetric, and its value is bounded in $[0, 1]$ when we use base 2 logarithm.

2.3.2 For Non-i.i.d. Random Variables

Random samples in some research topics are not independent and identically distributed. For this kind of data samples, we may want to find meaningful patterns or predict values from them. For example, in time series analysis, we may be interested in finding the property that changes constantly over time, or becomes invariant in the long run from the data samples. In the former aspect, we may want to find the trend or the seasonality of the sequence; in the latter one, we aim to decide whether the sequence is stationary or not. All these properties are dependent on the order of data samples in the sequence. These properties change if we permute these samples in the sequence. Because data samples are not i.i.d. and the order of samples is important, methods used to analyse i.i.d. samples are no longer appropriate to analyse sequence data. For example, sample variance does not capture the relationship between two adjacent samples in an ordered sequence, so the similarity in the sense of "order" will not be found by using this statistic. In addition, after randomly permuting samples in two sequences using the same permutation function, the covariance and the KL divergence will be the same, but the trend of these two sequences

may change. Therefore, it is necessary to use other similarity measures if we are dealing with sequence data.

Two kinds of similarity measures used to analyse time series data are given in [25]. The first kind of similarity measures consider the similarity of the shape, and another kind of similarity measures evaluate the similarity at the structural level. They are respectively called the shaped-based similarity and structure-based similarity measures [31].

Let $S_x = \{X_i\}_{i=1}^n$ and $S_y = \{Y_i\}_{i=1}^m$ be two time series. Assume m and n are equal. Euclidean distance, also known as the L2 distance, can be used as a kind of shape-based similarity measure that estimates the distance of S_x and S_y by using the following equation:

$$D(S_x, S_y) = \sqrt{\sum_{i=1}^{n}(X_i - Y_i)^2}$$

This method estimates the distance between two time series by considering the distance at the same time stamps. However, for two time series data that one of which is just a shifted sequence of other, Euclidean distance may fail to find that they are similar. Dynamic time warping (DTW) [5] is a method using dynamic programming approach to align two time series S_x and S_y. The time axis may be stretched or compressed to fit the two time series, and their length m and n can be different. For two time series which are only differ in their time stamps, it is possible to use DTW to find their relatedness.

Instead of comparing the shape, structure-based similarity measures extract global features from time series data, and use these features to measure the similarity of two time series data. We can compute statistics such as mean value, variance, skewness, and kurtosis from the data. One of the commonly used statistic is *auto-correlation*.

Let σ_x be the variances of time series S_x, auto-correlation function evaluates the dependency of the r-th and the s-th element in this sequence:

$$\frac{E[(X_r - \bar{X}_r)(X_s - \bar{X}_s)]}{\sigma_x^2}$$

Another statistic that evaluates two elements in different sequence data is *cross-correlation*. Cross-correlation function can be used to estimate the dependency of the r-th elements in S_x and the s-th element in S_y:

$$\frac{E[(X_r - \bar{X}_r)(Y_s - \bar{Y}_s)]}{\sigma_x \sigma_y}$$

Auto-correlation function and cross-correlation function measure the similarity of random variables in the sequences with time delay $t = |s - r|$. They measure the divergence of a sequence or two sequences at only two time points, rather than in a total view. These

two measures are the extension of correlation coefficient, and can be applied to sequence data.

One divergence measure based on KL divergence for two Markov sequences is Kullback-Leibler divergence rate, which is:

$$\lim_{n\to\infty} \frac{1}{n} D(P_x^n || P_y^n)$$

where P_x^n and P_y^n are the probability functions of a length-n sequence. This measure, however, can not be applied to any sequence data. It is shown that there exist two sequences such that $\lim_{n->\infty} \sup \frac{1}{n} D(P_x^n || P_y^n) \neq \lim_{n->\infty} \inf \frac{1}{n} D(P_x^n || P_y^n)$, which means that Kullback-Leibler divergence rate does not exist in [43].

2.3.3 Similarity Measures for Strings

String is a special type of sequence data that elements in it may not be numerical. Therefore, these elements may not be comparable. As a result, some measures in section 2.3.2 are not applicable. Before introducing functions to measure string similarity, we give some notations and terminologies of strings.

Let Σ be the set of alphabets. A **string** s is a sequence of alphabets $s_1 s_2 \ldots$ that $s \in \Sigma^*$. If a string is with length-n, then $s = s_1 s_2 \ldots s_n$. For such a string $s \in \Sigma^n$, we have $|s| = n$. A **null string** s is with $|s| = 0$. The **concatenation** of two strings u, v is uv, and their length $|uv|$ is $|u| + |v|$. Let x, u, v, and w be strings. If $x = uvw$, we say that u, v, and w are respectively the **prefix**, **substring**, and **suffix** of x. A string $t = t_1 t_2 \ldots t_m$ is a **subsequence** of s if there is a *strictly increasing function* $f : \{1, \ldots, m\} \to \{1, \ldots, n\}$ that $t_i = s_{f(i)}$. For example, "ca", "ct", and "at" are length-2 subsequences of string "cat", but "ac" is not. If t is a subsequence of s, we define the length of t in s to be $\ell_s(t) = f(m) - f(1) + 1$. For example, if s="cat", t_1="at", and t_2="ct", then $\ell_s(t_1) = 2$ and $\ell_s(t_2) = 3$.

If two strings are similar, it should be easy to transfer from one to the other. Under this idea, we can measure the divergence of two strings by considering the *cost* of transferring from one string to another, which is the edit distance of two strings. Besides the cost, we can also estimate *the amount of information* we need to transfer one string to the other as the distance. This kind of method is based on Kolmogorov complexity.

Kolmogorov complexity of string s, $K(s)$, is the length of the shortest binary program which computes s. $K(s,t)$ is the length of the shortest program that computes both s and t, as well as having a way to tell s, t apart. Conditional Kolmogorov complexity $K(s|t)$ is the length of the shortest binary program which computes s given t. By Kolmogorov complexity, information distance of two strings s and t, $E(s,t)$, can be defined as the length of the shortest binary program to transfer s to t and t to s. i.e.,

$$E(s,t) = \max\{K(s|t), K(t|s)\}$$

By normalizing the information distance, we have the normalized information distance (NID) [30]:

$$\mathrm{NID}(s,t) = \frac{E(s,t)}{\max\{K(s), K(t)\}}$$

Note that Kolmogorov complexity is a theoretical lower bound, and it is generally not computable. Therefore, the compression length of a string is used instead as an approximation. Let C be a compressor, and $C(s)$ denotes the length of compressing s using C. The compression distance of s and t using C is

$$E_c(s,t) = C(st) - \min\{C(s), C(t)\}$$

and the normalized compression distance (NCD) of s and t cab be defined using E_c and C [30]:

$$\mathrm{NCD}(s,t) = \frac{E_c(s,t)}{\max\{C(s), C(t)\}}$$

Another analogous similarity measure which is also based on data compression is proposed in [26]. This measure is called compression-based dissimilarity measure (CDM). The equation of CDM is

$$\mathrm{CDM}(s,t) = \frac{C(st)}{C(s) + C(t)}$$

CDM is close to 1 if s and t are not related, and is close to $\frac{1}{2}$ if s and t are similar. CDM will never be zero even when $s = t$.

2.3.4 Similarity Measures Using Kernel Method

Kernel methods have been successfully used in pattern analysis and machine learning. Given a domain $\mathcal{D}$, we can use a map ϕ that maps any element $d \in \mathcal{D}$ to a feature space $\mathcal{F}$ and perform inner product of two elements in $\mathcal{F}$ by defining a kernel function $k : \mathcal{D} \times \mathcal{D} \to \mathbb{R}$. Note that the inputs of k are elements in $\mathcal{D}$, so it is usually not necessary to explicitly define the mapping ϕ.

For example, if the objects we deal with are vectors, we can compute their similarity in their vector space. For two vectors $X = \{x_1, \ldots, x_n\}$ and $Y = \{y_1, \ldots, y_n\}$, we can use linear kernel function to estimate the similarity of X and Y, which is the inner product of X and Y, $\langle X, Y\rangle = \sum_{i=1}^{n} x_i * y_i$. For two non-zero vectors X and Y, $\langle X, Y\rangle = 0$ if and only if X and Y are orthogonal.

A normalized version of linear kernel is the well-known cosine similarity, which is cosine of the included angle of X and Y. Cosine similarity is defined as:

$$\frac{\langle X, Y \rangle}{||X||||Y||} = \frac{\sum_i x_i y_i}{\sqrt{\sum_i x_i^2}\sqrt{\sum_i y_i^2}} \tag{2.5}$$

In cosine similarity, two vectors are similar if they are similar in direction. The length of vectors is not taken into consideration. It can be seem that the function of correlation coefficient and cosine similarity are the same if $\bar{x} = 0$ and $\bar{y} = 0$ from equation 2.2 and 2.5.

Another well-known kernel function is radical basis function (RBF) kernel which is defined as:

$$k(x, y) = \exp\{-\frac{||x - y||^2}{2\sigma^2}\}$$

where x and y are two objects that $||x - y||$, mostly is Euclidean distance, is defined. It is easy to see that RBF kernel is maximized when $x = y$, and decreases when $||x - y||$ increases.

Given two sets of samples X and Y, let their feature maps be ϕ_x: $X \to \mathcal{F}$ and ϕ_y: $Y \to \mathcal{G}$ where $\mathcal{F}$ and $\mathcal{G}$ are Reproducing Kernel Hilbert Spaces (RKHS) with kernel functions k_x and k_y respectively. Given p_{xy}, the joint probability measure of X and Y, the Hilbert-Schmidt Independence Criterion (HSIC) [20], which is the square of Hilbert-Schmidt norm of the cross-covariance operator from $\mathcal{G}$ to $\mathcal{F}$, is defined as:

$$\begin{aligned}
&\mathrm{HSIC}(p_{xy}, \mathcal{F}, \mathcal{G}) \\
&= E_{x,x',y,y'}[k_x(x, x')k_y(y, y')] + E_{x,x'}[k_x(x, x')]E_{y,y'}[k_y(y, y')] \\
&-2E_{x,y}[E_{x'}[k_x(x, x')]E_{y'}[k_y(y, y')]]
\end{aligned}$$

where $(x, y) \sim p_{X,Y}$ and $(x', y') \sim p_{X,Y}$ are two independent pairs of random variables. $\mathrm{HSIC}(p_{xy}, \mathcal{F}, \mathcal{G}) = 0$ if and only if X and Y are independent.

It is also possible to define kernel functions for strings. By string kernels, we can evaluate the similarity of strings using kernel methods. Many string kernels for similarity measures can be found in [33]. We select some of kernel functions from this literature and introduce them below.

In p-spectrum kernel, we first define a feature map

$$\phi_u^p(s) = |\{(v_1, v_2) : s = v_1 u v_2)\}|, |u| = p,$$

which is the number of times the length-p string u occurs in s. The p-spectrum kernel is defined as:

$$k_p(s, t) = \sum_{u \in \Sigma^p} \phi_u^p(s)\phi_u^p(t)$$

If we give different weights $\sqrt{w_u}$ for each u above, and check all possible sub-strings, we will get another kernel function:

$$k(s,t) = \sum_{u \in \Sigma^*} \phi_u(s)\phi_u(t)$$

where $\phi_u(s) = \sqrt{w_u}|\{(v_1, v_2) : s = v_1 u v_2)\}|$.

If instead of sub-strings, we use sub-sequences to define the string kernel function, we can compare strings by inexact matching. Let u be a subsequence of s, we define a feature mapping:

$$\phi_u(s) = \sum_{v \in s, v=u} \lambda^{\ell_s(v)},$$

where $\lambda \in [0, 1]$ is a parameter. The length-p string subsequence kernel is

$$k_p(s,t) = \sum_{u \in \Sigma^p} \phi_u(s)\phi_u(t) = \sum_{u \in \Sigma^p} \sum_{v \in s, v=u} \sum_{v \in t, v=u} \lambda^{\ell_s(v)+\ell_t(v)}$$

For example, let s="cat" and t="cart", then $\phi_{\text{“}ct\text{”}}(t) = \lambda^4$. Common subsequences which can be used to compute the kernel function is “c”, “a”, “t”, “ca”, “ct”, “at”, and “cat”. In our example, if $p = 1$, $k_p(s,t) = 3\lambda^2$; if $p = 2$, $k_p(s,t) = \lambda^4 + \lambda^5 + \lambda^7$.

Chapter 3

Feature-based Knowledge Transfer Framework

The feature-based knowledge transfer framework is applied to find the mapping of features between two domains to transfer knowledge. We first define what is mapping between two sets:

Definition 1. *A mapping $\mathcal{M} : \mathcal{F} \to \mathcal{G}$ is a binary relation between two feature sets $\mathcal{F}$ and $\mathcal{G}$. In addition, if $(f_i, g_j) \in \mathcal{M}(\mathcal{F}, \mathcal{G})$, we say feature $f_i \in \mathcal{F}$ and $g_j \in \mathcal{G}$ are **mapped together**.*

In our framework, sets $\mathcal{F}$ and $\mathcal{G}$ are feature sets in two domains. We compute the mapping $\mathcal{M}$ between $\mathcal{F}$ and $\mathcal{G}$. Knowledge is transferred based on this mapping. Let $\mathcal{F} = \{f_1, f_2, \ldots, f_S\}$ and $\mathcal{G} = \{g_1, g_2, \ldots, g_T\}$ be two sets of features in source and the target domains respectively. Assume $T \leq S$. A mapping $\mathcal{M} : \mathcal{G} \to \mathcal{F}$ can be defined (If $T \geq S$, we define $\mathcal{M} : \mathcal{F} \to \mathcal{G}$), and a matrix C can be computed given $\mathcal{M}$:

$$C_{i,j} = \begin{cases} 1 & \mathcal{M}(g_i) = f_j, \\ 0 & \text{otherwise} \end{cases}$$

We compute the new feature sets from original feature sets of the two domains and the matrix C, and instead of using the original feature sets $\mathcal{F}$ and $\mathcal{G}$ to learn and test the model, we use new feature sets $\tilde{\mathcal{F}}$ and $\tilde{\mathcal{G}}$. The elements $\tilde{f}_i$'s and $\tilde{g}_j$'s in the two new feature sets have the following property: $\tilde{f}_k = f_j$ and $\tilde{g}_k = g_i$ if and only if $C_{i,j} = 1$. That is, only those features in G that map to any non-empty element in F, and only those features in F that are mapped by any non-empty element in G, are used in training and testing procedures. The correspondence of these features can be found from matrix C. Therefore, given C or $\mathcal{M}$, we can compute and use $\tilde{\mathcal{F}}$ as the new feature set of the source domain dataset and $\tilde{\mathcal{G}}$ as the new feature set of the target domain dataset.

We compute $\mathcal{M}$ and the corresponding matrix C according to the divergence/similarity between any two features in different domains. What is the *divergence*/*similarity* of features? Generally speaking, measuring similarity is a process of estimating the *distance* between two objects in a problem domain. The data type of objects can be random variables, structured data, images, strings, etc. Distance between objects reflects their degree of similarity. In this work, objects are features, and features are extracted from sensors. Therefore, similarity of a feature is affected by the similarity of sensors that it is extracted from. Generally speaking, if the information two features provides is highly correlated in the sense of predicting the label, we can say that the two features are similar. Therefore, in activity recognition datasets, when two features are extracted from the same set of sensors or different sensors that can provide analogous events, for example microwave and toaster for the event of preparing the meal, we say they are similar.

As a result, we say that feature f_i and g_j are similar in the task of inferring the label ℓ in the label set L if the two features provide highly correlated information to ℓ. We can decide the mapping only if we can measure the relatedness of the information. Here is a definition for similarity:

Definition 2. *Let $h_L : R \times S \to \mathbb{R}$ be a function that can measure the relatedness of the information two elements in R and S provide to L. For three elements $r_i \in R$ and $s_j, s_k \in S$, if $h_L(r_i, s_j) < h_L(r_i, s_k)$, we say r_i is more similar to s_j than to s_k, or the divergence between r_i and s_j is less than the divergence of r_i and s_k.*

By the definition above, we define the feature similarity:

Definition 3. *Let $f_i \in \mathcal{F}$ and $g_j \in \mathcal{G}$ be two features in two datasets, and L is the set of labels. If there is a function $h_L : \mathcal{F} \times \mathcal{G} \to \mathbb{R}$ as we define in definition 2 that $h_L(f_i, g_j) \simeq \min_k h_L(f_k, g_j)$, we say that f_i is similar to g_j in the feature set $\mathcal{F}$ under the task of inferring L, or briefly f_i and g_j are similar without ambiguity. If $f_i = \operatorname{argmin}_k h_L(f_k, g_j)$, then f_i is the most similar element to g_j.*

In our framework, if f_i and g_j are highly similar, we want to map these two features together. Note that if there is more than one possible choice of function h, it is possible that the result of these functions is not consistent. The choice of the similarity measure function h_L may affect the result of deciding which two features are similar, and therefore make the computed mapping different. It would be necessary to identify some properties or criteria to choose h_L instead of considering all possible choices.

In the following sections, we explain the details of the proposed framework. An overview of this framework is given in figure 1.1. The feature reformulation procedure is used to make each feature provide different information, and the feature alignment procedure is used to automatically compute the feature mapping. These two procedures

depend on feature similarity which is assumed to be available or computable. The feature similarity procedure can provide necessary information to the two procedures: In feature reformulation procedure, it may need to provide the information about whether two features in the same datasets are identical or not; in feature alignment procedure, it needs to give the estimation of divergence of features in two datasets.

The purpose of this feature-based knowledge transfer framework is to align similar features in different domains before a learning algorithm starts to train a model. Since this framework is run when preprocessing datasets, it can be applied to any existing learning algorithms after extracting features. Although the application we aim at is building activity recognition models in an intelligent environment, this knowledge transfer framework should be available for more general usage. Knowledge can be transferred between two domains by applying this framework as long as we have similarity of features.

3.1 Feature Similarity

In definition 3, we defined the meaning of similar features. In our framework, it is necessary to estimate the similarity between any two pairs of features between the source and target domain if it is not given. The problem of definition 3 is that the function h_L is not easy to define. Different functions are only suited to various scenarios.

In this section, we only discuss some general issues of measuring similarity of features from different datasets. The specific methods that we have applied to estimate feature similarity will be discussed in chapter 4.

3.1.1 Issues on Measuring Feature Similarity

The adopted similarity measure in this framework should be able to measure the relatedness of the information a feature can provide to infer the label. Therefore, it is necessary to take this requirement into consideration. Moreover, it would be better if the similarity measure can be related to the learning model. For example, a similarity measure for sequence data should be used with a learning model for time series data, and when a learning model based on the distribution of the data is used, similarity measures based on their probability distributions would be more suitable.

Measuring similarity of features in different datasets is challenging. A reason is that we usually do not have a dataset that contains both these features at the same time. Namely, it would be very difficult to estimate the joint probability of them from datasets directly without any assumption. As an example, recall the mutual information of random

variables X, Y:

$$I(X;Y) = \sum_{X,Y} p(X,Y) \log \frac{p(X,Y)}{p(X)p(Y)}$$

We can see from this equation that if we can not compute $p(X,Y)$, estimating $I(X;Y)$ is not possible.

Defining an appropriate similarity measure case by case may be necessary. According to whether the labelled target domain data is available or not, we proposed different methods to estimate the feature similarity in this work. If the target domain data are available, we can estimate feature similarity from data. In this case, the function h_L in definition 3 and 2 we chose in our experiment is expected Jenson-Shannon divergence. Otherwise, we defined sensor profiles from background knowledge, and use them to estimate the distance of features. These methods will be described in chapter 4.

3.2 Feature Reformulation

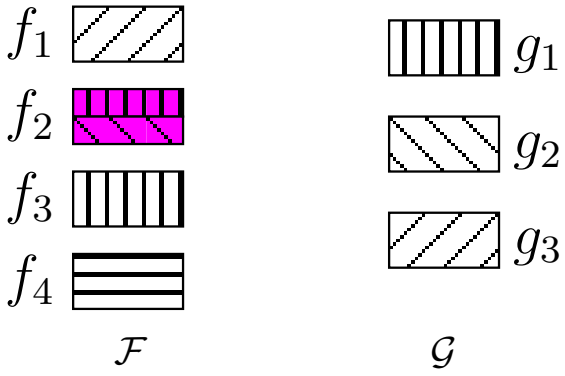
The first procedure in this framework is feature reformulation. This procedure should be executed before the feature alignment procedure because the feature divergence used in the feature alignment procedure will be estimated again after reformulating the features. This procedure is necessary when we have one of the following situations:

1. One feature f_i contains various information about more than one features in $\mathcal{G}$
2. More than one features in $\mathcal{F}$ provide similar information to a feature in $\mathcal{G}$

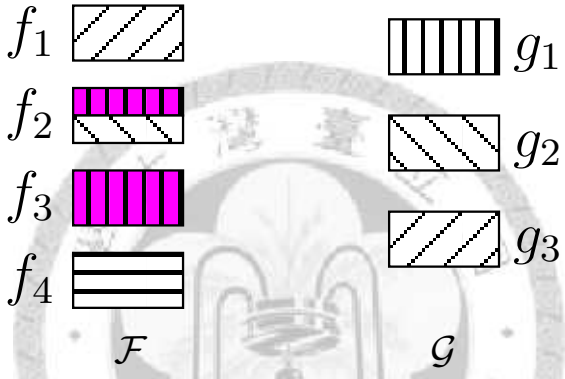
In figure 3.1, we show two examples that feature reformulation is needed. In the situation of figure 3.1(a), if we only map f_2 and g_2 together, we lose some useful information. It would be better to separate feature f_2. Similarly, in the situation of figure 3.1(b), we do not want to just map f_3 and g_1 together and lose the information that f_2 can provide. Ideally, we want to separate the information that f_2 can provide in to two parts, and merge one of them to f_3 to make all features provide highly divergent features as well as do not lose any information, as shown in figure 3.2. Therefore, the ideal feature reformulation procedure implementation should have the following two properties:

1. Make features provide highly divergent information to infer the label.
2. Information loss caused by the feature reformulation procedure should be minimized.

We will introduce our feature reformulation implementation for the two scenarios mentioned in section 1.1.1. Our implementation, however, does not satisfy all the two properties.



(a) One feature contains various information.



(b) Two features provide similar information.

Figure 3.1: Two situations that feature reformulation is necessary.

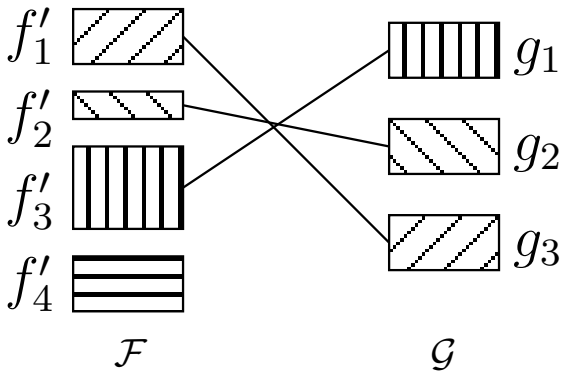


Figure 3.2: An ideal situation after reformulating features.

3.2.1 Feature Reformulation by Using Data Samples

In information theory, entropy is used to describe the amount of information in random variables. Let $\{X_i\}_{i=i}^n$ be the random variables of features, and Z be the random variable of labels. The amount of information these features contain can be expressed by Shannon entropy $H(X_1, \ldots, X_n)$ if X_i's are discrete [42]. For continuous random variables, $h(x_1, \ldots, h_n)$, the differential entropy, can be used. Note that $H(X_1, \ldots, X_n) \leq \sum_{i=1}^n H(X_i)$ and $h(X_1, \ldots, X_n) \leq \sum_{i=1}^n h(X_i)$ with equality if and only if X_i's are independent, no matter they are discrete or continuous random variables [12].

If a labelled dataset is available, we compute the transformation L_A with transformation matrix A from data. L_A can be linear or non-linear transformation. In this work, we focus on the linear case. Let I denote the mutual information function in information theory, and $\mathbf{X} = [X_1, \ldots, X_n]^T$. The two properties we given above can be expressed by the following two equations:

1. $H(\mathbf{X}) = H(A\mathbf{X})$, and
2. $\forall i, j$, $I(L_A(X_i); L_A(X_j)|Z) = 0$,

where X_i and X_j are respectively random variables of features f_i and f_j in the same dataset.

The two equations above means the Shannon entropy of these features before and after applying the linear transformation is the same. Also, given the label, any two features in the same dataset provide independent information to the label. Note that it does not seem to be possible to find a unique transformation which can make the mutual information of two features given the label to be zero, since the joint distributions of the random variables may change given different labels. An alternative is to minimize the value of the metric that we choose to evaluate the dependency of these random variables.

A possible solution to implement the transformation is Principal component analysis (PCA) [24]. PCA is a well-known method which transforms the original dataset by finding a new basis to express the original data samples. In the linear case, PCA can find an orthogonal basis in the original vector space. Because of this property, the random variables are uncorrelated after applying PCA. PCA can guarantee that elements in the converted feature set are *uncorrelated* but not necessary *independent*, since independent random variables are uncorrelated, but uncorrelated random variables may not be independent in general cases.

Another candidate is Independent component analysis (ICA) [10], which is often used in signal processing. ICA aims at finding statistically independent components from mixtures of signals. Independent components in ICA methods can be found by the following three strategies [23]:

- Maximize nongaussianity of components
- Maximize likelihood function
- Minimize mutual information

Unlike PCA which minimizes the correlation between each components, ICA minimizes the mutual information. PCA and ICA are the solutions to make the features contain uncorrelated or independent information. However, they are not applicable to the feature reformulation procedure in our framework. The reason is that both PCA and ICA are unsupervised methods. That is, they do not take the label into consideration. Therefore, we can not use these two methods directly in the feature reformulation procedure in our framework.

If we relax the constraints and find *uncorrelated* components instead of independent ones, we can make use of the theory in PCA to define a linear transformation. Let $\det(A)$ denote the determinant of a square matrix A, and $|\det(A)|$ the absolute value of this determinant. Before we show our implementation when labelled datasets are available, we first show that regardless of what the linear transformation function is, we can easily verify whether the first property of the ideal transformation is satisfied or not from its transformation matrix. Specifically, given a linear transformation L_A, we show that $H(\mathbf{X}) = H(A\mathbf{X})$ if A is invertible. For continuous random variables, $h(\mathbf{X}) = h(A\mathbf{X})$ if $|\det(A)| = 1$.

Lemma 3.1. *Let $\{X_i\}_{i=i}^n$ be the random variables of features, $(X_1, \ldots, X_n)$ be a random vector in vector space U, and $(\mathcal{X}_1, \ldots, \mathcal{X}_n)$ be a random vector in vector space V. If a linear transformation $L_A : U \to V$ whose transformation matrix A is invertible, $H(A\boldsymbol{X}) = H(\boldsymbol{X})$ for discrete random variables X_i's. If X_i's are continuous random variables, $h(A\boldsymbol{X}) = h(\boldsymbol{X})$ when $|\det(A)| = 1$.*

Proof. First we show the case of discrete random variables. By the definition of Shannon entropy, $H(X_1, \ldots, X_n) = -\sum_{X_i} P(X_1, \ldots, X_n) \log P(X_1, \ldots, X_n)$. Because A is invertible, L_A is one-to-one and onto. For any $s \in V$ that $L_A(r) = s$, we have

$$P(s) = \sum_{r:L_A^{-1}(s)=r} P(r) \tag{3.1}$$

where L_A^{-1} is the inverse function of L_A. Because A is invertible, for any s there is one and only one $r \in U$ that $L_A^{-1}(s) = r$, and thus $P(s) = P(r)$. Therefore,

$$H(X_1, \ldots, X_n) \;=\; -\sum_r P(r) \log P(r) \tag{3.2}$$

$$= -\sum_{s=L_A(r)} P(s)\log P(s) \tag{3.3}$$

$$= -\sum_{s\in\mathcal{X}} P(s)\log P(s) \tag{3.4}$$

$$= H(\mathcal{X}_1,\ldots,\mathcal{X}_n) \tag{3.5}$$

On the other hand, if X_i's are continuous random variables, we have

$$h(A\mathbf{X}) = h(\mathbf{X}) + \log|\det(A)| \tag{3.6}$$

$$= h(\mathbf{X}) \tag{3.7}$$

Equation 3.6 is given in [12]. □

Now we show how to modify the method in PCA and give a linear transformation method. In PCA, the covariance matrix of features is used to find the a new basis to represent the dataset. The new basis consists of the eigenvectors of the covariance matrix. Take into consideration the labels, let $\Sigma = \mathrm{Cov}(\mathbf{X}|Z)$, L_A be the linear transformation with transformation matrix A, and $A\mathbf{X} = [\mathcal{X}_1,\ldots,\mathcal{X}_n]^T$. We have

$$\mathrm{Cov}(A\mathbf{X}|Z) = A\Sigma A^T = \Lambda, \tag{3.8}$$

where Λ is a diagonal matrix. Similar to PCA, we have $\forall\, i,j,\ \mathcal{X}_i \perp \mathcal{X}_j|Z$.

From equation 3.8, we know that if we can compute $\mathrm{Cov}(\mathbf{X}|Z)$, we can use the same method as what is used in PCA. However, $\mathrm{Cov}(\mathbf{X}|Z)$ is a function of Z, not a constant. Therefore, we compute $E[\mathrm{Cov}(\mathbf{X}|Z)]$ instead. Let Σ_x denote $E[\mathrm{Cov}(\mathbf{X}|Z)]$ and Λ be a diagonal matrix. We have

$$E[\mathrm{Cov}(A\mathbf{X}|Z)] = E[A\,\mathrm{Cov}(\mathbf{X}|Z)A^T] = A\Sigma_x A^T = \Lambda \tag{3.9}$$

From equation 3.9, we know that A is the matrix of the linear transformation we want to find. If Σ_x is not a diagonal matrix, we compute the eigenvectors of Σ_x and normalize them such that the Euclidean norms of these vectors are all 1. In this case, the row vectors of A are these normalized eigenvectors. On the other hand, if Σ_x is a diagonal matrix, then let A=I, the identity matrix. From this procedure, we have the following property:

Lemma 3.2. *A is invertible, and $|\det(A)| = 1$.*

Proof. If $A = I$, we have done. In the case of $A \neq I$, because Σ_x is a real symmetric matrix, its eigenvectors are orthogonal. Since the Euclidean norm of these eigenvectors are 1, the determinant of A is ± 1. So A is invertible, and $|\det(A)| = 1$. □

Now the reason why Shannon entropy of $\{X_i\}_{i=i}^n$ before and after applying this linear transformation L_A is unchanged is obvious. In addition, the expected covariance of any two features after applying L_A is 0. We have the following theorem:

Theorem 3.1. *There is no information loss after transferring the data by applying a linear transformation with a transformation matrix whose determinant is 1. In addition, the expected covariance of any two transferred features given the label is 0.*

Proof. Because A is invertible and $|\det(A)| = 1$, from lemma 3.1 there is no information loss. Moreover, the expected covariance of any two features is 0 because $\Lambda = E[\mathrm{Cov}(A\mathbf{X}|Z)]$ is a diagonal matrix. □

As the conclusion, $\mathbf{X}^T A^T$ is the vector of random variables of the new feature set whose Shannon entropy is the same with $\mathbf{X}^T$. Given a labelled dataset, we can use matrix A to linearly transfer it without losing any information as well as minimizing the expected covariance of any two features.

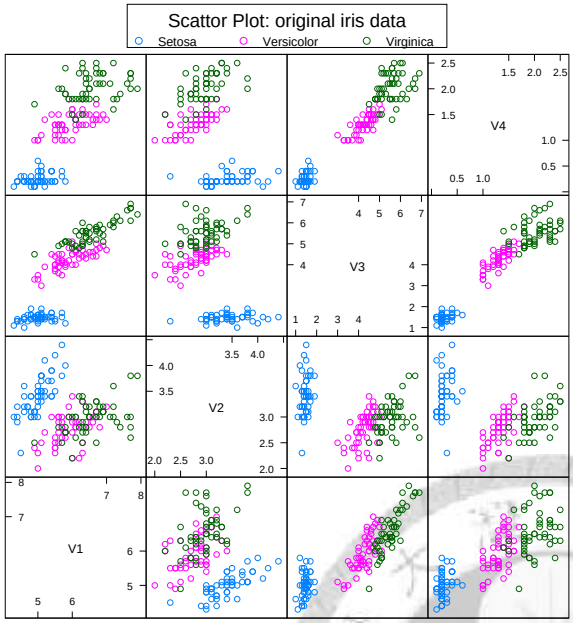
Two transformation functions for the source and target domain are respectively computed from their datasets before executing the next procedure of our framework. We run a simple experiment to show the effect of applying this linear transformation on the well-known iris dataset comes from UCI Machine Learning Repository[17]. We compare the distribution before and after applying this linear transformation procedure and PCA using scatter plots in figure 3.3. We can find from figure 3.3(c) that after applying PCA, data samples are evenly distributed in global view, while in most cases our linear transformation method makes data samples evenly distributed in the same class.

3.2.2 Feature Reformulation by Profiles

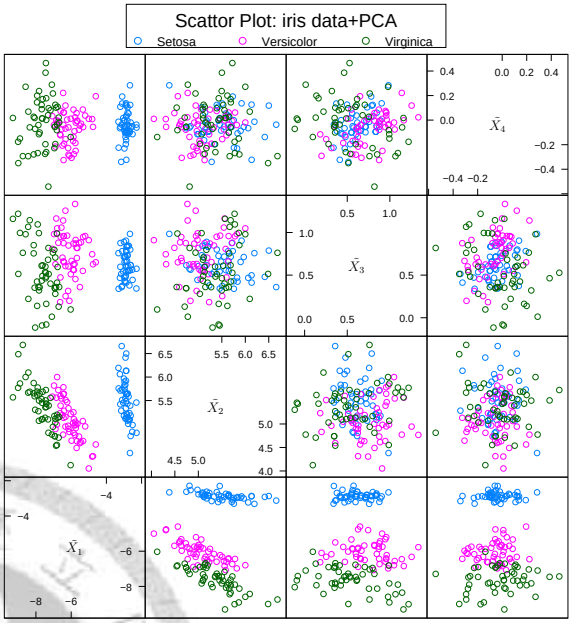
Recall the scenario we give in section 1.1.1 that we have no target domain dataset. In this case, it is impossible to estimate the similarity of features in the same dataset from labelled data. In order to merge analogous information, we use the background knowledge about the features.

Specifically, we encode background knowledge of sensors as sensor profiles, which is a length-N binary string that describes sensors. N is the number of properties. The binary string is used to represent the properties of a sensor, and can reflect the information provided by different sensors. If the profiles of two sensors are similar, these sensors should provide analogous information to infer the activity, and features extracted from these sensors should also be similar. We define four types of properties to be included in the sensor profile as follows:

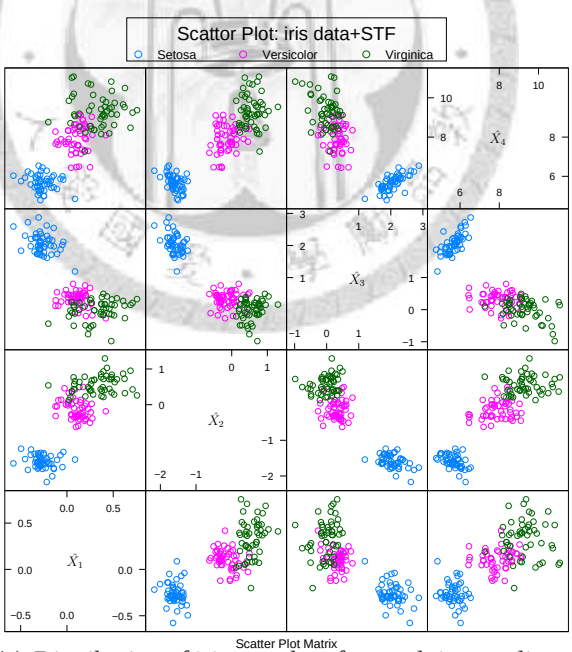
- Object: On which object the sensor is set up.
- Location: The location of the object.
- Sensor type: What kind of sensor it is.



(a) Original iris dataset



(b) Distribution of iris samples after applying PCA



(c) Distribution of iris samples after applying our linear transformation

Figure 3.3: Comparing the distribution of data samples before and after applying PCA and our linear transformation method. STF in figure 3.3(c) stands for "Supervised Trans-Formation".

- Event: What event the sensor detects.

If a sensor is related with the i-th property, the i-th binary value in the profile will be 1. For example, a sensor deployed on the microwave in the kitchen is related with properties "kitchen" and "microwave", so there will be respectively a value 1 assigned for these two properties in the profile. This method is inspired by [49] and [50] that these properties is similar to the meta features in these two papers.

Besides the properties, we also assigned a non-negative importance value for these properties. The importance value indicates the degree of assurance about whether an activity is performed or not given the state of a sensor which has this property. These properties and their importance values are all defined by hand in our work. A detailed list of these properties along with their importance values we used will be given in chapter 4.

We compute feature profiles from sensor profiles. Let $P_{s_i} = \{p_1^i, \ldots, p_N^i\}$ be the profile of sensor i, and $P_{f_j} = \{q_1^j, \ldots, q_N^j\}$ be the profile of feature j. If feature j is extracted from a set of sensors A_s, we have

$$q_k^j = \bigvee_{S_i \in A_s} p_k^i, k = 1, 2, \ldots, N \tag{3.10}$$

$\bigvee$ is logical operator OR. For example, if feature i is extracted from sensors S_i and S_j with profiles $\{1, 1, 0, 0\}$ and $\{1, 0, 1, 0\}$ respectively, $P_{f_i} = \{1, 1, 1, 0\}$.

Using feature profiles, we can run the feature reformulation procedure even when the labelled dataset is not given. We combine the data of features whose profiles are identical. Recall that we define the profile to reflect the information that a sensor can provide to infer activities. Sensors with different profiles should provide different information, so the feature extracted from these sensors will also provide different information. Features provide identical information to infer activities if and only if they have identical profiles. Therefore, we combine features with identical profiles in order to merge the same information together.

Note that in this case, we can not guarantee that different features in the new feature set provide unrelated information. We may also loss information because we apply logical operator OR to combine features with identical profiles. Under the scenario that labelled dataset is not available, these two properties of the ideal transformation function may not be satisfied. Nevertheless, it is still effective to used this method in our framework. We verify it by showing the experimental results under this scenario in chapter 4.

3.3 Feature Alignment

After the feature reformulation procedure, we compute the feature mapping by feature alignment procedure. This procedure finds similar feature pairs (f_i, g_l) that f_i and g_l are

in different datasets. The purpose of this procedure is not only to make the size of the two feature sets in source and target domain datasets be the same but also to find the correspondence between any two features in two domains to transfer knowledge in the best way.

3.3.1 Graph Matching Algorithms

Given a similarity measure, we formulate the problem by defining a weighted graph, and apply graph matching algorithms to compute the mapping. Specifically, we reduce our problem to minimum cost perfect matching problem or stable marriage problem in graph theory. In the following sections, we will introduce the two matching problems. After that, we show how to reduce our mapping problem to graph matching problems.

Matching

In graph theory, a matching is a set of edges that any two edges can not share one vertex. Given a graph $G(V, E)$, a **matching** of G is a subset of E that if two edges (v_i, v_j) and (v_r, v_s) are in the subset, $i \neq r, s$ and $j \neq r, s$. A **perfect matching** E_p is a matching that no vertex is left behind in the matching. That is, for each $v_i \in V$, we have one and only one edge $e \in E_p$ that $e = (v_i, v_j)$ or (v_j, v_i).

Minimum Cost Perfect Matching

A **minimum cost perfect matching** [14] is a perfect matching with minimum cost. That is, given a weighted graph G, it finds a matching in the graph G such that the summation of the weights of these edges in the matching is minimized. Formally, let w_i be the weight of edge e_i, the perfect matching E_p is a set of edges in a perfect matching E_k with the property:

$$E_p = \underset{E_k}{\operatorname{argmin}} \sum_{e_i \in E_k} w_i \tag{3.11}$$

Stable Marriage

The stable marriage problem [18] in graph theory is a problem of finding a stable matching between two sets of vertices, V_a and V_b. A matching is **stable** if when an edge (a_i, b_i) is in the matching, there is no edge (a_j, b_j) in the matching such that a_i prefers b_j to b_i and b_j also prefers a_i to a_j. Note that a stable matching may not have minimum total cost, as the example we show in figure 3.4.

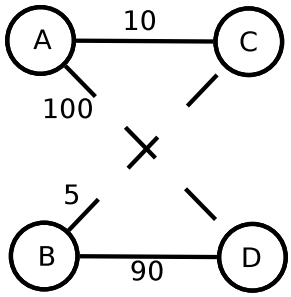


Figure 3.4: A stable marriage matching (A, D), (B, C) is with cost 105, which is not a minimum cost matching.

3.3.2 Feature Mapping by Graph Matching

We can reduce our feature mapping problem to the graph matching problem. Assume we have two sets of features A and B. There are m features in A and n features in B. We define a complete bipartite graph $K_{m,n} = (U, V, E)$ that $|U| = m$ and $|V| = n$. Vertex $u_i \in U$ and $v_j \in V$ represent a feature $a_i \in A$ and $b_j \in B$ respectively. We also assign weight values to all edges in E. The weight value of edge (u_i, v_j) is the divergence of feature a_i and feature b_j. If u_r and v_s are matched in the graph $K_{m,n}$ according to the algorithm, feature a_r and feature b_s are mapped together, as we defined in definition 1. By this reduction, we can solve our feature mapping problem by solving the graph matching problem.

Choosing different graph matching algorithms has different meanings for knowledge transfer. Observing figure 3.4, we can see the difference. If the algorithm for minimum cost perfect match problem is applied, we are going to find a mapping in global view, namely, the total divergence between two feature sets after mapping is minimized. On the other hand, the stable marriage algorithm aligns the most similar features in two datasets first. In this case, total divergence may not be optimal, but the most preferred pairs between two features will not be sacrificed.

Note that in our method, some features in the datasets may be ignored because of the following two reasons. First, if $m \neq n$, the matching computed by the graph matching algorithm is not perfect. If u_i is not covered in the matching, its corresponding feature a_i will be ignored when we transfer knowledge. Besides, some of these edges in the matching may be with high weight values, which means the corresponding features are in fact not similar. In this case, it would be better to ignore these edges with high weight values in the matching.

3.3.3 Measuring Divergence of Datasets

If there is more than one source domain available, intuitively we should transfer knowledge between two "similar" domains instead of two highly divergent domains. The relationship between divergence of two datasets and performance of models has been studied in [2], [48], and [9], which show that divergence of two datasets and model performance of knowledge transfer are related. However, currently there is no standard method to evaluate the divergence between two different domains to decide how to transfer knowledge. There is also no known criteria on how to choose a method to estimate the divergence of two datasets in transfer learning.

It may be possible to extend the feature similarity measure to estimate divergence between datasets in two domains according to our feature-based knowledge framework, as proposed in [9]. Let $\mathcal{F}$ and $\mathcal{G}$ be two feature sets in two domains, from definition 1, we can use the following equation to estimate the divergence of two datasets:

$$\mathbb{D}_{\mathcal{F},\mathcal{G}}(\mathcal{F};T) \approx \sum_{(f_i,g_j)\in\mathcal{M}(\mathcal{F},\mathcal{G})} \mathbb{D}_{\{f_i\},\{g_i\}}(f_i;T)$$

where T is the task of domains, and $\mathbb{D}_{\mathcal{F},\mathcal{G}}(\mathcal{F};T)$ is the estimated distance between $\mathcal{F}$ and $\mathcal{G}$ under T. That is, the total distance of two domains is the summation of the distance of each features which are mapped together under the task.

Therefore, assume the divergence between features in two datasets are given or computable. We can compute an one-to-one correspondence of these features between the source and target domain dataset by applying the graph matching algorithms. The summation of feature divergence in the mapping may be used to estimate the distance of source and target datasets. We may use minimum cost perfect match algorithm to compute the best matching between nodes in a bipartite graph. The result of this matching can give a lower bound of divergence of two datasets.

Chapter 4

Experiments

In this chapter, we describe the knowledge transfer experiment conducted according to our framework under two scenarios. The details of activity recognition datasets, including the datasets we used, the algorithms of data preprocessing and feature mapping, parameters, and the results are given. The feature reformulation procedures for the two scenarios are that we described in section 3.2. We measure our knowledge transfer framework by the accuracy of models.

Recall the two scenarios we gave in section 1.1.1.

1. A dataset collected in a laboratory environment is available, and we are preparing to deploy sensors to the target domain environment.

2. A dataset collected in a laboratory environment is available, and we have also collected and labelled some samples in the target environment.

In these two scenarios, data samples collected in the laboratory environment is our source domain data. We proposed the following solutions for these two scenarios, and apply our framework to run the experiments:

1. We transfer knowledge by sensor profiles.

2. We use labelled source and target domain data samples to transfer knowledge.

In our experiments, each feature is extracted from only one sensor, so feature profiles and sensor profiles are identical. We will use *sensor profile* and *feature profile* interchangeably in the following sections without confusion.

Table 4.1: The number of features and activities in the datasets

Dataset	MAS S1	MAS S2
♯ of sensors	76	70
♯ of activities	23	25

4.1 Datasets and Data Preprocessing

The datasets we used come from the course of MAS622J, Pattern Recognition and Analysis, at MIT[1]. There are two datasets in the MIT MAS622J dataset (we call them MAS S1 and MAS S2), which are collected from two different single-person apartments. Only ambient sensors are installed in the two apartments, so there is no sensor on human body, and no video, image, and voice data in the two datasets. More information about these datasets can be found in [44] and [45].

For state change sensors in these datasets, we find the start and end time sensors being triggered. By doing this, we have the state of all sensors in each time stamps. Each data sample is a list of all sensors with binary values indicating which sensors are triggered at that time. We then convert the sensor data using a fixed-length time interval. The length of the time interval in our experiment is 30 seconds without overlapping. If a sensor is triggered at any time stamp in a time interval, its value in that time interval will be set to 1. In addition, if there are multiple activities performed in a time interval, we extract only the latest one. For example, if in the raw data, there is an activity sequence $\{i, i, (ij), (ij), j\}$, the extracted sequence will be $\{i, i, j, j, j\}$. The number of features and activities extracted from the raw data of these datasets are shown in table 4.1. Note that the number of features and activities in table 4.1 may not be the same with what is described in the papers we listed above because some sensors are ignored if they are not triggered at all time stamps. In addition, there are some samples with no activity annotated in the raw data. We set a new NoAct label for these samples.

4.2 The Feature Reformulation Procedure

We use the method described in section 3.2.1 and 3.2.2 to reformulate the feature sets first in our experiments. In the first scenario, we use sensor profiles to reformulate features. Sensor profiles are encoded according to background knowledge to give the information of how similar two sensors in the same dataset are. Features with identical profiles in the same dataset are merged. Properties of the sensor profile, including the type and

[1]available: http://courses.media.mit.edu/2004fall/mas622j/04.projects

Table 4.2: Sensor properties and their importance values

Type	Property	Importance Value
location	bathroom	4
	bedroom, kitchen, living room, toilet	2
	study room, balcony, outside, gateway, Other private space, Other public space	1
object	shower faucet, flush, stove/oven/burner, dishwasher, washing machine/clothes dryer, telephone, tea/coffee machine, Audio/Video Equipment	4
	washbasin/sink faucet, garbage disposal	3
	cupboard, microwave, toaster, food grain, light	2
	closet/cabinet, cutlery, drawer, window, door, chair/sofa, bed, refrigerator/freezer, fan, box, container	1
sensor	switch	2
	motion/PIR	1
event	entering/leaving/moving, operate appliances, get/put/find something, something is on the object, light on/off	1

importance values defined in our experiments, are given in table 4.2.

In the second scenario, we compute two expected matrices from labelled data samples in the two datasets respectively, and use these two matrices to decide the linear transformation of the two datasets. Specifically, let $\{X_i\}_{i=i}^{n}$ be random variables of features, Z be the random variable of labels, and define $\mathbf{X} = [X_1, X_2, \ldots, X_n]$. We first compute $\Lambda = E[\mathrm{Cov}(\mathbf{X}|Z)]$, and then compute the eigenvectors of Λ. These eigenvectors are normalized such that their Euclidean distance are all 1. The matrix used to run the linear transformation is consist of these normalized eigenvectors as the row vectors. In this case, $E[\mathrm{Cov}(X_i, X_j|Z)]$ which is computed from labelled data samples is the "divergence" of X_i and X_j in the same dataset.

4.3 Estimate Feature Similarity

The difference between using background knowledge and data samples to transfer knowledge is mainly on how we estimate the feature similarity after the feature reformulation procedure: When some data samples are available, we use them to estimate feature similarity. Otherwise, feature similarity is estimated from background knowledge. The features similarity measure is used to measure the divergence of information that features can provide to infer the label. Two features are similar if they provide similar information about the label.

4.3.1 Feature Similarity Estimation by Using Data Samples

In this work, when some labelled target domain data samples are given, the method we used to estimate the divergence between two features is based on Jenson-Shannon divergence which is introduced in equation 2.4 in chapter 2. In order to take labels into consideration, we separately compute the Jenson-Shannon divergence between features for data samples with different labels, and summarise these results to estimate these features' overall Jenson-Shannon divergence. Specifically, let P_i and Q_j be the distribution of $f_i \in \mathcal{F}$ and $g_j \in \mathcal{G}$ respectively, we compute the *expected* Jenson-Shannon divergence of two distributions P_i and Q_j given label Z, which is:

$$E[\mathrm{JSD}(P_i(f_i|Z)||Q_j(g_j|Z))] = \sum_k f_t(z_k)\,\mathrm{JSD}(P_i(f_i|Z = z_k)||Q_j(g_j|Z = z_k)), \quad (4.1)$$

where f_t is the probability distribution function of the label Z in the dataset.

Therefore, $\mathrm{JSD}(P_i(f_i|Z = z_k)||Q_j(g_j|Z = z_k))$, Jenson-Shannon divergence of f_i and g_j with different z_k is separately computed. Then $E[\mathrm{JSD}(P_i(f_i|Z)||Q_j(g_j|Z))]$, the

overall Jenson-Shannon divergence, can be estimated. We use $E[\text{JSD}(P_i(f_i|Z)||Q_j(g_j|Z))]$ as the divergence of features $f_i \in \mathcal{F}$ and $g_j \in \mathcal{G}$. That is,

$$\mathbb{D}(i,j) = E[\text{JSD}(P_i(f_i|Z)||Q_j(g_j|Z))] \tag{4.2}$$

The advantage of using Jenson-Shannon divergence is that it is symmetric, its value is bounded, and it is not necessary to estimate the joint probability of two features to estimate their divergence.

4.3.2 Feature Similarity Estimation by Profiles

If the data sample in target domain dataset is not available, we still need to estimate $\mathbb{D}(i,j)$ between features f_i and g_j in our framework. We estimate it using the profiles defined in section 3.2.2. If two sensors or features have almost identical profiles, they are similar in our method. Measuring feature similarity by their profiles is to draw an analogy between two features.

The similarity measure in this scenario is a two-step procedure. For each pair of features in two datasets, we first compute the divergence for each type of profile property. Afterwards, we compute the summation of the divergence of these types of profile property. Specifically, let $\mathbb{T}$ be the set of profile property types, and w_k^t be importance value of the k-th property which belongs to the type-t property in the profile. We compute the divergence of the type-t property between two features using their profiles with the following equation:

$$\mathbb{D}_t(i,j) = \frac{\sum_{p_k \in \mathbb{T}, p_k^i \neq p_k^j} w_k^t - \sum_{p_k \in \mathbb{T}, p_k^i = p_k^j = 1} w_k^t}{\sum_{p_k \in \mathbb{T}, p_k^i \neq p_k^j} 1 + \sum_{p_k \in \mathbb{T}, p_k^i = p_k^j = 1} 1} \tag{4.3}$$

And estimate the total divergence as:

$$\mathbb{D}(i,j) = \sum_t \mathbb{D}_t(i,j) \tag{4.4}$$

It is easy to see that for any two features with identical profiles, the positive term $\sum_{p_k^i \neq p_k^j} w_k$ will be 0. Besides, if two features have many different properties, they will have a large feature divergence.

4.4 The Feature Alignment Procedure

After reformulating feature sets and computing the feature divergence between any two features in the source and target domain, we can compute the mapping of features by graph matching algorithms. A complete bipartite graph $K_{m,n}(U,V,E)$ is defined, and

$\mathbb{D}(i,j)$ is used as the weight of the edge (u_i, v_j). The problem of computing feature mapping is reduced and solved by a graph matching problem. Details of the problem reduction is described in section 3.3.2.

Feature alignment procedure is simple. Regardless of what method is used to estimate divergence between features, as long as $\mathbb{D}(i,j)$ is available, we can just apply a graph matching algorithm to compute the feature mapping. The graph matching algorithm we used in all our experiments is the stable marriage algorithm. We adopt this algorithm because of the following two reasons. First, the matching computed in our experiments is actually not "perfect"[2]. In addition, according to the Hall's theorem [21], perfect matching may not exist in some cases. By adding some pseudo-nodes and pseudo-edges to the bipartite graph, we can solve these two problems. However, since the feature mapping computed by minimum cost perfect matching algorithm and stable marriage algorithm in the experiment of the first scenario happens to be identical under our profile setting, we just reduce our feature mapping problem to stable marriage problem instead of the minimum cost perfect matching problem in the second scenario either. The experimental result of mapping features in the first scenario is given in table 4.3. Sensors in the same row of table 4.3 will be aligned with in the experiment of the first scenario.

Note that some rows in the table are combinations of several "identical" sensors, which means their profiles are the same. We only show one of these combined features here in the table to make this table readable. We can see that most of these rows are reasonable. TV(141) in MAS S2 is aligned with DVD(56) in MAS S1 because they are all related to the event "operate appliances". On the other hand, some rows in table 4.3 seem to be illegitimate. For example, aligning TV(101) with Jewelry box(139) does not make sense. However, we can see that these two features are with a larger divergence value, which means that they are actually not very similar. Therefore, we know that it is necessary to decide a threshold value in the feature alignment procedure to eliminate bad matches in the matching computed by the graph matching algorithm.

Before we show and discuss our experimental results, we want to argue that defining sensor profiles is a feasible solution to transfer knowledge. Recall that we focus on building a activity recognition model in the smart home environment that only has ambient sensors in it. Since deploying sensors in an environment requires knowledge about the environment and these sensors, background knowledge is not difficult to obtain. Observe the properties we list in table 4.2, which is the profile we use in our experiments. Object, location, and sensor types are all easy to known. In addition, since sensors are deployed according to the task, we usually have expectation of what kind of event a sensor can detect. Dataset preprocessing procedures in activity recognition, including feature extraction

[2] A trivial explanation: In a bipartite graph $G(U,V,E)$, if $|U| \neq |V|$, we can not find a perfect matching.

Table 4.3: The mapping of features computed by stable marriage algorithm using MAS662J dataset. In the parentheses are sensor IDs in the original dataset. The weight values are the weight of the edges in the matching.

MAS S2	MAS S1	Weight
Light switch (109)	Light switch (101)	-9
Shower faucet (130)	Shower faucet (93)	-8
Sink faucet - hot (100)	Sink faucet - hot (68)	-7
Light switch (102)	Light switch (107)	-7
Light switch (106)	Light switch (108)	-7
Toilet Flush (112)	Toilet Flush (100)	-7
Light switch (119)	Light switch (105)	-7
TV (141)	DVD (56)	-7
Light switch (103)	Light switch (104)	-6
Light switch (107)	Light switch (92)	-6
Burner (117)	Burner (94)	-6
Toaster (108)	Toaster (131)	-5
Microwave (115)	Microwave (143)	-5
Medicine cabinet (127)	Medicine cabinet (57)	-5
Garbage disposal (84)	Garbage disposal (98)	-5
Cabinet (104)	Cabinet (132)	-4
Drawer (114)	Drawer (125)	-4
Containers (124)	Containers (60)	-4
Refrigerator (66)	Refrigerator (126)	-4
Hamper (78)	Cabinet (67)	-4
Drawer (126)	Drawer (82)	-3
Door (85)	Door (140)	-3
Door (137)	Door (130)	-2.286
Door (134)	Door (141)	-2
Door (51)	Door (54)	-2
TV (101)	Jewelry box (139)	0
Sink faucet - cold (91)	Window (136)	0
Door (133)	Closet (81)	2
Telephone (69)	Cabinet(133)	2.5
Stereo (122)	Cabinet (85)	4

and feature selection, require background knowledge about sensors in the environment. We have to know the specification of sensors attached on an object to interpret the raw data and find the state of this object. Therefore, background knowledge we required to define the sensor profiles is necessary not only in our framework but also in conventional activity recognition problems.

4.5 Experiments and Results

We apply our feature-based knowledge framework to the two scenarios given in section 1.1.1. Since the assumption of the first scenario is that we do not have any target domain data sample, we do not generate any variation on the dataset by applying cross-validation to train the model. On the other hand, we run a 10-fold cross-validation experiment for the second scenario. In the experiment of this scenario, target domain dataset is randomly separated into 10 parts, and nine of them are combined to estimate the feature similarity with the source domain data. The remaining one part is used to test the accuracy of the model. We trained activity recognition models using only the source domain dataset to find out if the knowledge transfer framework can successfully extract and transfer useful knowledge. We also trained activity recognition models using both the source domain dataset and nine of the 10 parts of the labelled target domain data samples to simulate the situation of when we apply this framework under the second scenario in a smart home system. The non-transfer learning experiment using the same feature set is also performed as the baseline experiment to verify the effectiveness of our framework.

After finishing the feature alignment procedure, feature sets in two datasets have the same cardinality, and an one-to-one mapping relationship between features in different datasets is also available, which is the key point in our knowledge transfer framework. Since we have an one-to-one mapping relationship of features in two domains, we can apply ordinary activity recognition algorithms to build models. In all our experiments, we use libsvm [8] with RBF kernel to train and test models. We only use default values for all libsvm parameters in our experiment.

4.5.1 Knowledge Transfer by Using Data Samples

The result of knowledge transfer given some target domain samples is shown in figure 4.1(a) and 4.1(b). The x-axis gives the number of features used to transfer knowledge, and the y-axis in the left side shows the accuracy. Feature pairs with divergence larger than the threshold value will not be used to train and test the model, even though they are mapped together in the feature alignment procedure. The black lines in the two

figures give the relation between the threshold value and the number of selected features, which can be used to find the accuracy of models under different threshold values. Setting a higher threshold value results in having more features in our experiments, which also means being more tolerant to use feature pairs which may be highly divergent.

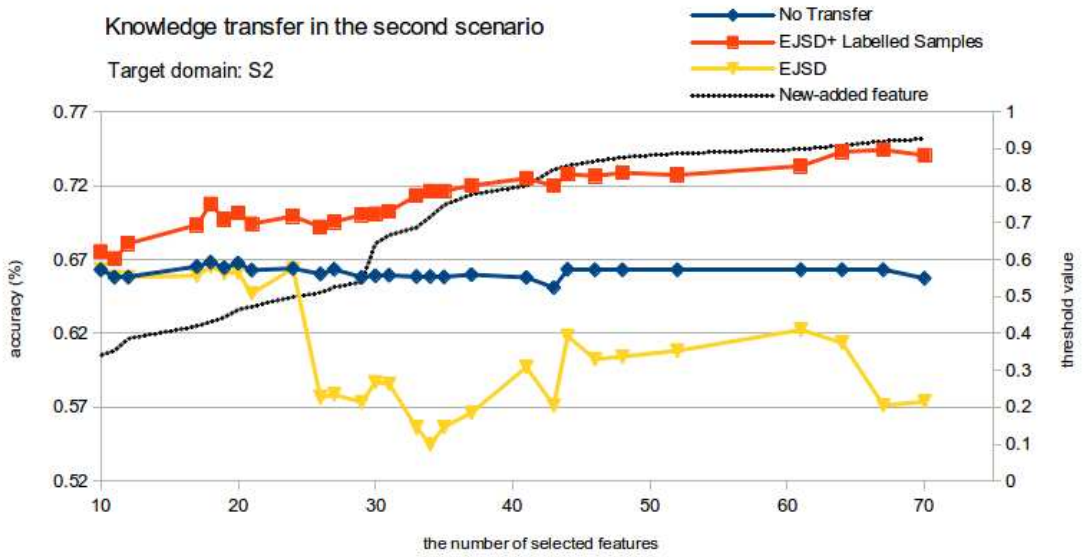
The two yellow lines in figures are the result of using expected JSD as the divergence measure, and training the model only using the labelled source domain data samples; the orange lines give the result of using expected JSD as the divergence measure, and using both the labelled source and target domain data samples to train the model. The blue lines are the baseline experiments which are non-transfer learning. We run 10-fold cross-validation experiments in this scenario. Therefore, we give the error bar of experimental results in figure 4.2, 4.3, and 4.4.

These experiments show that with labels, we can estimate the feature similarity. Expected Jenson-Shannon divergence can be a valid divergence measure of features in our framework. There is a catch, however, that we need to choose a good threshold value. By observing these figures, it can be found that with different threshold values, the experimental results vary wildly. We can see that the model performs bad at some points in the yellow lines, while some other points on the yellow are very closed to the non-transfer learning results. Therefore, threshold value plays an important role when we train a model in our framework. On the other hand, using labelled target domain data samples on hand, the model can perform better than non-transfer learning. Therefore, we can say that our knowledge transfer framework can help to extract and appropriately transfer knowledge between different domains.

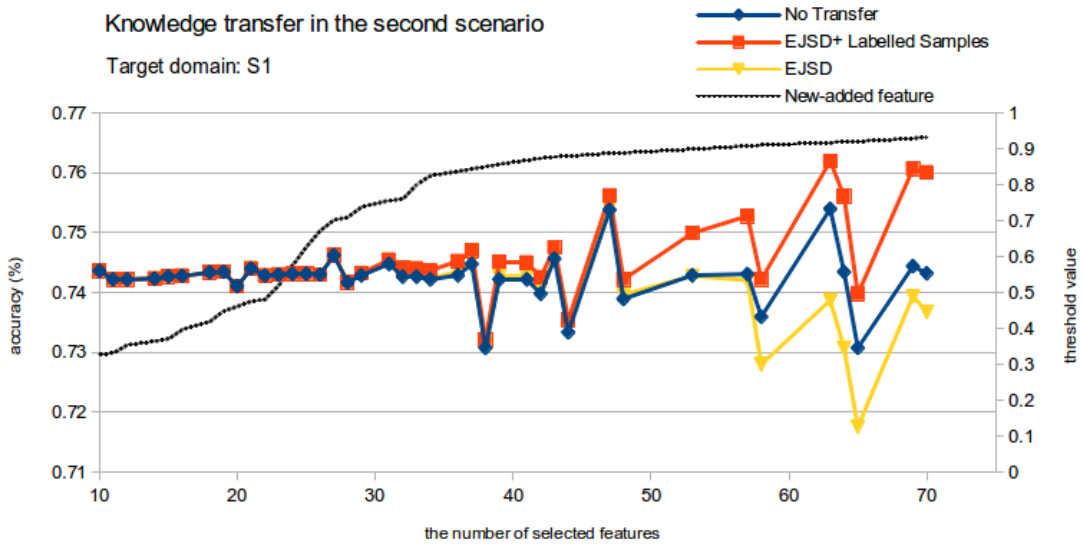
4.5.2 Knowledge Transfer by Profiles

We show the experimental results of transferring knowledge by profiles between MAS S1 and MAS S2 in figure 4.5(a) and 4.5(b). In figure 4.5(a), we use MAS S1 as the source domain, and MAS S2 is the target domain. In figure 4.5(b), we exchange the role of the two datasets. As the experimental results in previous section, the x-axis of these figures gives the number of selected features in our experiments, and the y-axis is the accuracy of the model. The number of features in this experiment is smaller than that in previous experiments because some features in the same dataset are merged in the feature reformulation procedure in this scenario.

On the orange line, we show the result of transfer learning that uses sensor profiles with importance values; on the green line, we show the result of transfer learning that uses sensor profiles whose importance values are all set to 1. These two lines show the accuracy of models trained according to our framework. The blue line, indicating supervise learning with importance values, and the yellow line, indicating supervise learning

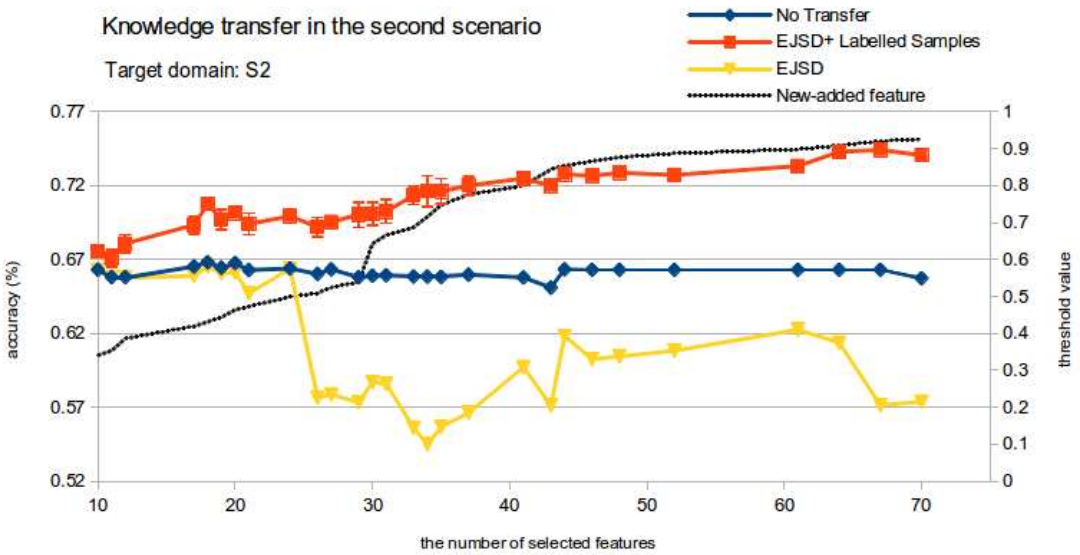


(a) Knowledge transfer from MAS S1 to MAS S2 by using data samples

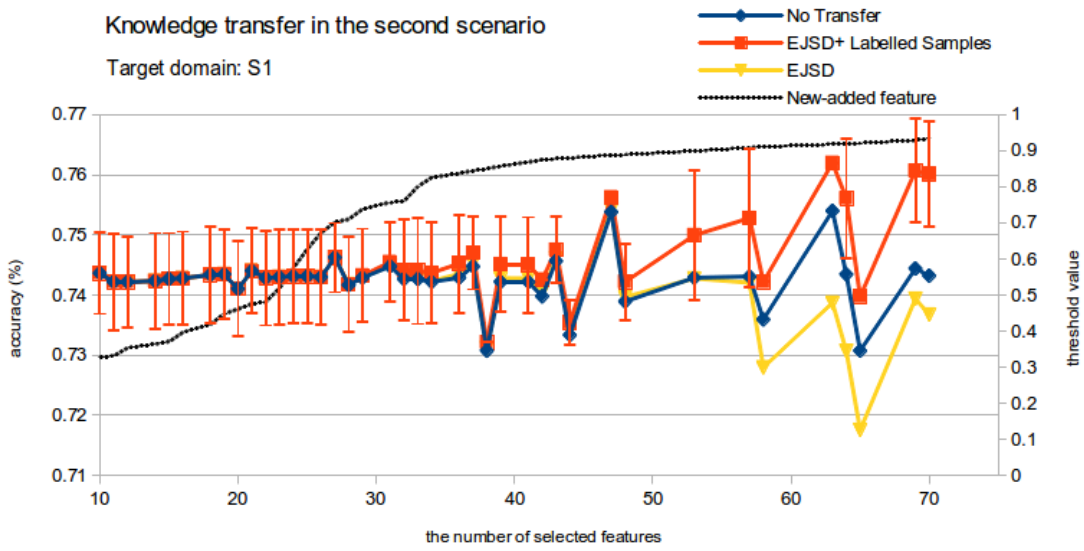


(b) Knowledge transfer from MAS S2 to MAS S1 by using data samples

Figure 4.1: Knowledge transfer between MAS S1 to MAS S2 given a labelled source domain dataset and some labelled target domain data samples. The x-axis is the number of features. The y-axis in the left side is the accuracy in %, and the y-axis in the right side is the threshold values. This y-axis with the black line shows the relationship of threshold values and the number of selected features.

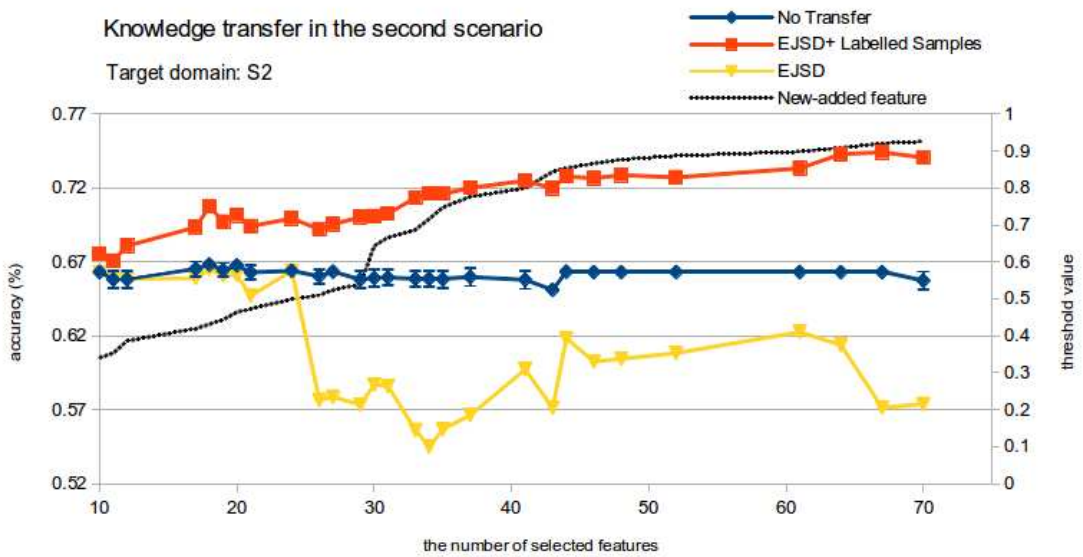


(a) Knowledge transfer from MAS S1 to MAS S2

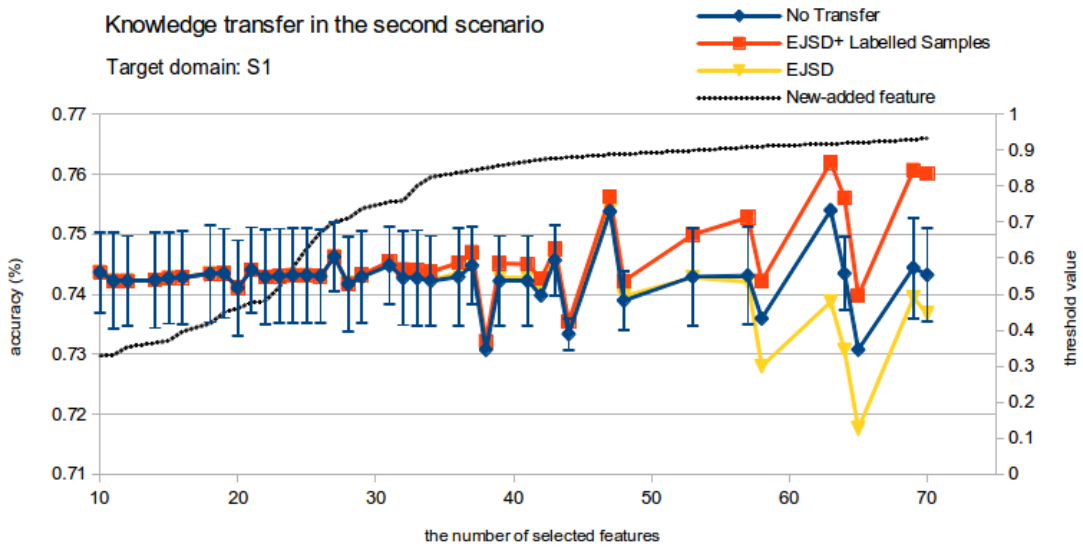


(b) Knowledge transfer from MAS S2 to MAS S1

Figure 4.2: Knowledge transfer between MAS S1 to MAS S2 given a labelled source domain dataset and some labelled target domain data samples. The error bar in these figures gives the standard deviation of the accuracy on the experiment of using all available data samples to train the model.

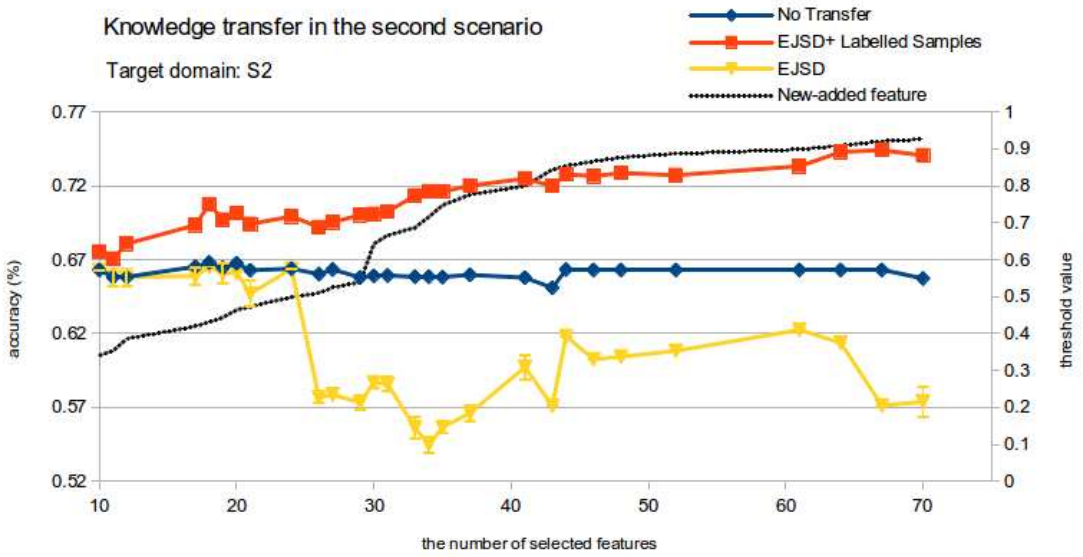


(a) Knowledge transfer from MAS S1 to MAS S2

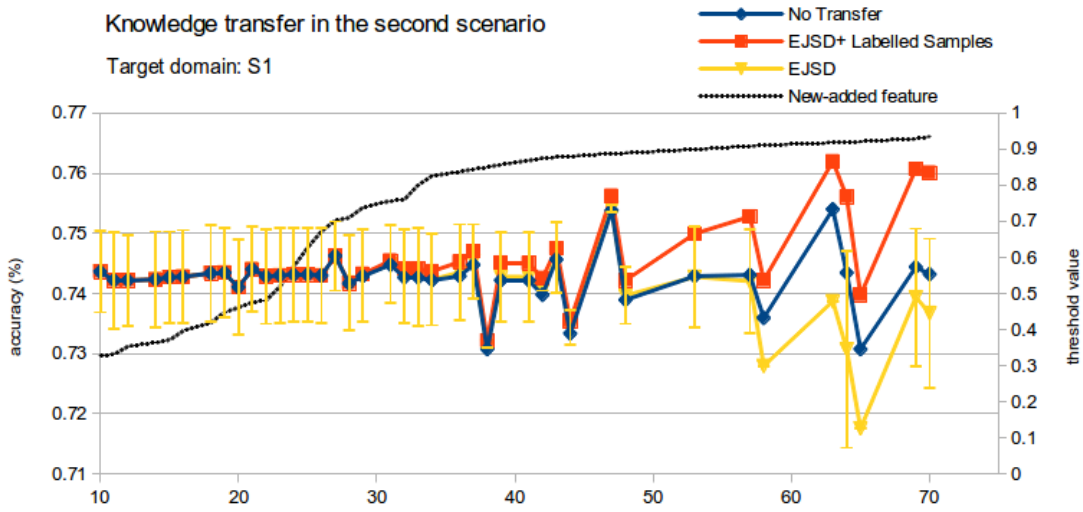


(b) Knowledge transfer from MAS S2 to MAS S1

Figure 4.3: Knowledge transfer between MAS S1 to MAS S2 given a labelled source domain dataset and some labelled target domain data samples. The error bar in these figures gives the standard deviation of the accuracy on the experiment of the non-transfer experiment.



(a) Knowledge transfer from MAS S1 to MAS S2



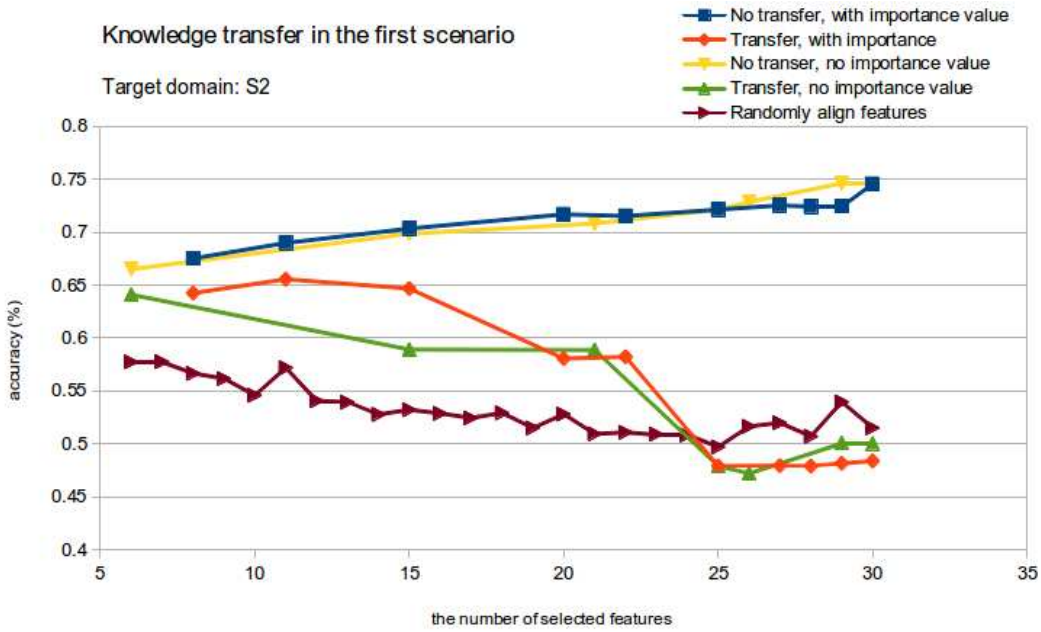
(b) Knowledge transfer from MAS S2 to MAS S1

Figure 4.4: Knowledge transfer between MAS S1 to MAS S2 given a labelled source domain dataset and some labelled target domain data samples. The error bar in these figures gives the standard deviation of the accuracy on the experiment of using only source data samples to train the model.

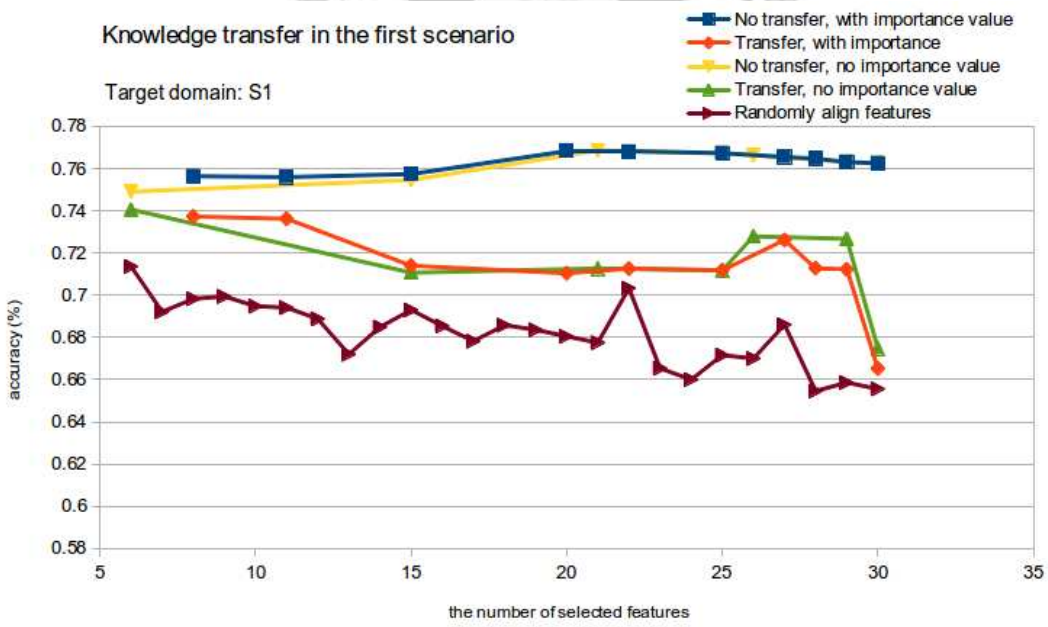
without importance values, are the results that run five-fold cross validation on only the target domain data. Therefore, they are non-transfer learning experiments. The meaning of the "with importance values" and "no importance values" descriptions is that their feature sets are respectively the same with those in the orange line and the green line. Finally, the brown line, indicating randomly align features, is the result of running the same training and testing procedures by using feature sets which are randomly chosen and mapped together. The blue, yellow, and brown lines are used as the baseline to verify whether sensor profiles can help to extract and transfer knowledge or not. Note that it is assumed that there is no target domain data samples in this scenario, so in the orange and green line, models are only trained using source domain data samples.

From these figures, we can say that this knowledge transfer framework is valid. Comparing to the result of randomly mapping features, encoding the background knowledge about the sensors and environment by sensor profiles definitely transfers useful knowledge. Similar to the previous experiments, threshold values play an important role on the performance of models. In figure 4.5(a), the models trained by randomly mapping features outperform the models trained by applying our knowledge transfer framework at some points. However, this phenomenon does not disprove the validity of our framework. On the contrary, it is an evidence showing that the feature divergence estimated from sensor profiles is valid. Observing figure 4.5(a), we can find that the result of random mapping performs better only when almost all feature pairs in the feature mapping are used to train the model. Therefore, we know that using highly divergent feature pairs to transfer knowledge can cause negative transfer, while using only feature pairs which are similar can successfully transfer useful knowledge in our framework. Feature divergence estimated from the sensor profiles is meaningful.

From the experiment in this scenario, we can say that it is possible to train a preliminary model whose performance is close to the best one by applying our framework. Therefore, when we want to install a smart home system to an environment, this preliminary model can be embedded into the system before this system starts to run. The difficulty of doing this is that we have to appropriately choose a threshold value. A small number of features may not contain sufficient information to train a good model, while selecting all feature pairs in the feature mapping may cause negative transfer. Our experiments, however, can not give an answer to this problem. We can only say that it may be better to select a smaller number of features as the initial parameter to train a preliminary activity recognition model.



(a) Knowledge transfer from MAS S1 to MAS S2 by sensor profiles



(b) Knowledge transfer from MAS S2 to MAS S1 by sensor profiles

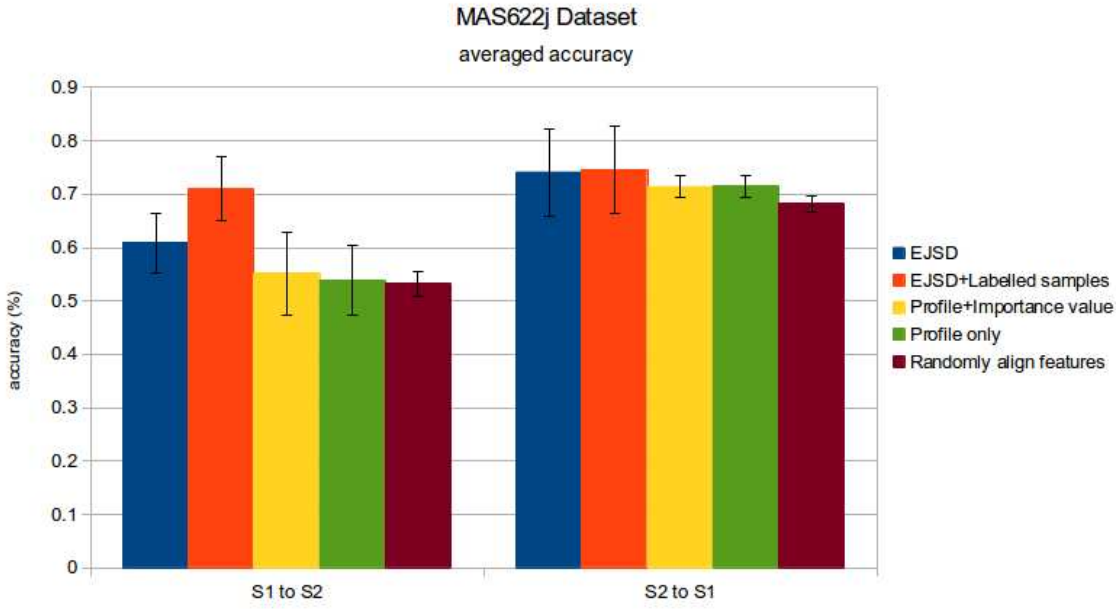
Figure 4.5: Transfer knowledge between MAS S1 and MAS S2 given the source domain dataset and background knowledge. The x-axis is the number of selected features and y-axis is the accuracy in %. The results of applying our framework (orange and green lines) are shown along with that of using the same feature set but trained and tested the model on only the target domain dataset (blue and yellow lines), and the result of randomly choose and align features (brown lines).

4.5.3 Further Analysis

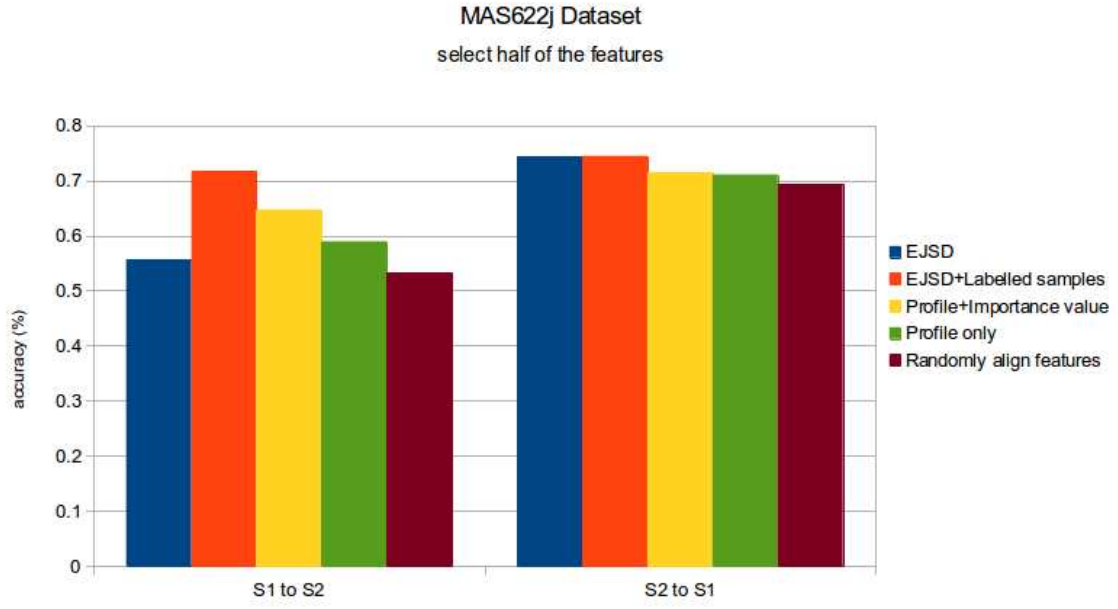
Finally, to further investigate the experimental results, we compare all knowledge transfer results by showing the averaged accuracy values of them in figure 4.6(a). Because of the dilemma of preserving knowledge as much as possible to transfer and preventing negative transfer, we also show the result of an intuitive solution to set the threshold value–selecting only half of the feature pairs whose feature divergence are smaller in figure 4.6(b).

In figure 4.6(a) and 4.6(b), S1 to S2 on the left side is the experiment of using MAS S2 as the target domain dataset, and S2 to S1 is that of using MAS S1 as the target domain dataset. Figure 4.6(a) gives the averaged accuracy of all the experimental results we shown in previous sections, with standard deviation shown as the error bar. Figure 4.6(b) is the result of using only half of feature pairs. Specifically, in figure 4.6(b), we select 35 features in the result of "EJSD" and "EJSD+Labelled samples", and only 15 features are selected in the others to train and test the model.

We can again verify the effectiveness of our feature-based knowledge transfer framework by the averaged accuracy values given in figure 4.6(a). Models that transfer knowledge by using sensor profiles and labelled target domain data samples outperforms the model trained from random mapped feature sets. Generally speaking, our framework leads to better performance when some labelled target domain data samples are given, and setting importance values on the property of sensor profiles can make the model performance better than not setting them.



(a) Averaged accuracy of all results



(b) The accuracy of choosing only half of features to transfer knowledge

Figure 4.6: Comparing the results under different scenarios. In figure 4.6(a), all accuracy values are used to compute the averaged value. In figure 4.6(b), only half of the features in the mapping are used.

Chapter 5

Conclusion

In this work, we proposed a feature-based knowledge transfer framework. Given beforehand or by estimating the similarity or divergence between features in source and target domain datasets, this framework can be used to automatically compute a new feature set of two domains to transfer knowledge from the source domain to the target domain. This knowledge transfer framework can be applied to create a preliminary activity recognition model in an intelligent environment under different scenarios.

We discussed two scenarios in this work. In one scenario, only the labelled source domain dataset is available. In the other scenario, labelled datasets from both source and target domains are both given. We gave possible solutions to both scenarios, and conducted our experiments on a public dataset to show the effectiveness of this framework. In the first scenario, background knowledge of two domains is encoded as sensor profiles to reformulate and estimate the divergence of any two features in two domains. The mapping relationship between features in different datasets is computed. In the second scenario, we estimated the divergence of any two features from labelled data samples, reformulated the feature sets by a PCA-like method, and computed feature mapping by adopting a graph matching algorithm. The same graph matching algorithm is used to compute a new feature set to transfer knowledge. The experimental results indicate that this feature-based knowledge transfer framework is valid.

5.1 Discussion

Now we discuss our work and the experiments. We also give the advantages and limitations of our method, and list possible future work.

5.1.1 About the Experiments

In our experiment of the first scenario, knowledge is transferred by sensor profiles since there is no target domain data samples. We defined properties of sensors, and assigned importance values to these properties. We used these properties to describe sensors as their profiles. The divergence of two features can be estimated from sensors that they are extracted from using these sensors' profiles. In the second scenario, since there are some labelled target domain data samples, we use expected Jenson-Shannon divergence as the divergence measure of features. Stable marriage algorithm is used in our experiments of both scenarios to compute the feature mapping to transfer knowledge.

Note that we are not arguing that sensor profiles and the importance values we defined for these datasets are the best settings. In additional, expected Jenson-Shannon divergence is just one of the possible divergence measures. It is possible to define different sensor profile settings or another divergence measures in our framework to train a model that outperforms ours. It is also possible to find different methods instead of feature profiles to encode background knowledge of sensors to evaluate similarity between features. All what we need is a measure to estimate the relatedness of information two features can provide to infer the performed activity. The implementation of these procedures in our framework can be various. The purpose of conducting these experiments is to show that knowledge transfer by our feature-based knowledge transfer framework is possible.

5.1.2 Limitations

One of the limitations in our framework is that we still have to set some parameters by hand. For example, we have to define sensor profiles from background knowledge when target domain data samples are not available. As mentioned in [22], defining sensor profiles manually is the limitation of this kind of methods. In addition, finding an appropriate threshold value to avoid negative transfer is also necessary. In our experiments, we only found that a smaller threshold value may prevent divergent information from being transferred, but it may also block off useful knowledge to be transferred. We only gave a trivial setting of selecting the best half of feature pairs in our experiments, but we can not say that this is the best choice in all cases. How to decide the best threshold value remains an open problem.

In addition, the proposed framework in this work does not deal with the difference between two label sets. Therefore, if the sets of activities performed in the source and target domains are highly different, this framework can not discovery new activities which are not in the source domain. Moreover, properties of sensors and sensor profiles we defined in this work are not universal. It is applicable only for activity recognition datasets

using only ambient sensors. For other activity recognition datasets using wearable sensors or cameras in the environment, it is necessary to define more properties in the profiles. In other problems such as transferring knowledge between different corpora in natural language processing, one has to define new profiles with totally different properties and importance values to apply this framework.

Finally, our framework can not evaluate whether transferring knowledge between the two given domains is appropriate or not. As mentioned in section 3.3.3, divergence between two domains can affect model performance. Our framework can not raise an alarm to an inappropriate knowledge transfer task.

5.1.3 Advantages

The advantage of this work is that the feature-based knowledge transfer framework is a general solution that it can be applied as long as similarity of features between two datasets is given or computable. Feature mapping can always be automatically computed. Based on our framework, one can train an activity recognition model by collecting a dataset in a laboratory environment which has sufficient number of sensors and various activities. After that, apply our framework according to the scenario: If there is no target domain data sample, define sensor profiles to transfer knowledge. If some labelled target domain samples are available, estimate similarity of features from data samples to train the model.

Another advantage of our work is that we have shown that we can use only source domain data samples to train the model in our experiments. In this case, knowledge is transferred by domain knowledge of sensors, which is encoded in sensor profiles. Most previous work that use transfer learning in activity recognition problem requires at least some unlabelled target domain data samples. Only some of the researchers proposed methods which do not need any data in the target domain. An example for this kind of knowledge transfer can be found in [22]. Comparing to the cost of data collection and data annotation, defining sensor profiles by hand should be relatively affordable. As we argued in section 4.3.2, the knowledge we need to create sensor profiles is not difficult to obtain.

5.2 Future Work

As the future work, we want to focus on extending the limitations we listed above. First, it is necessary to find a method to automatically compute the parameters such as threshold values and settings of sensor profiles. We may compute the threshold value from datasets

by cross-validation if both labelled source and target domain datasets are available. In the scenario that labelled target domain data samples are not available, however, learning all these parameters from only source domain datasets may suffer from the divergence of the two domains and cause negative transfer. A method to utilize both the source domain dataset and background knowledge to learn these parameters is necessary to further improve the availability of our framework.

Moreover, similarity functions can affect the result of feature mapping. Therefore, the choice of divergence measures can be critical to model performance in our framework. For example, in one of the scenarios, we compute expected Jenson-Shannon divergence between two features as the feature divergence used in our experiment. However, it does not seem likely that expected Jenson-Shannon divergence an appropriate divergence measure for all transfer learning tasks. It would be useful if we can define the properties or criteria which are critical in selecting the divergence measure.

People may want to transfer knowledge between not only the same but also different problem domains. For example, transfer the knowledge of rating of users between not only DVDs but also from DVDs to books. It will be very interesting to improve the framework such that it can transfer knowledge between different problem domains, or at least give a caution to a knowledge transfer task which may cause negative transfer. A difficulty for this is that in transfer learning, negative transfer is a concept of inappropriate knowledge transfer. To the best of our knowledge, however, there is no specific definition of negative transfer. Except for the result we showed in our experiment that the model performed worse than randomly mapping features, we do not know how to exactly decide whether a knowledge transfer task causes "negative transfer" or not. Therefore, giving a concrete definition to negative transfer and extend our framework to warn of inappropriate knowledge transfer will be very useful.

Finally, it does not seem to be feasible to define a universal profile of features for different learning tasks. However, automatically deciding the properties in profiles for different learning tasks may be possible. When the learning task is given, mining relative information to decide these properties from other knowledge sources such as the web may be a possible extension for this work.

Bibliography

[1] J. Baxter. A model of inductive bias learning. *Journal of Artificial Intelligence Research*, 12:149–198, March 2000.

[2] S. Ben-David, J. Blitzer, K. Crammer, and F. Pereira. Analysis of representations for domain adaptation. In *Advances in Neural Information Processing Systems 19*, pages 137–144, Cambridge, MA, 2007. MIT Press.

[3] S. Ben-David and R. S. Borbely. A notion of task relatedness yielding provable multiple-task learning guarantees. *Machine Learning*, 73(3):273–287, Jan. 2008.

[4] S. Ben-David, T. Lu, T. Luu, and D. Pál. Impossibility theorems for domain adaptation. *Journal of Machine Learning Research – Proceedings Track*, 9:129–136, 2010.

[5] D. J. Berndt and J. Clifford. Using dynamic time warping to find patterns in time series. In *KDD Workshop*, pages 359–370, 1994.

[6] B. Cao, S. J. Pan, Y. Zhang, D.-Y. Yeung, and Q. Yang. Adaptive transfer learning. In *AAAI*, 2010.

[7] R. Caruana. Multitask learning: A knowledge-based source of inductive bias. In *Proceedings of the 10th International Conference on Machine Learning*, pages 41–48. Morgan Kaufmann, 1993.

[8] C.-C. Chang and C.-J. Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011.

[9] Y.-t. Chiang, W.-C. Fang, and J. Y.-j. Hsu. Knowledge source selection by estimating distance between datasets. In *Proceedings of the 2012 Conference on Technologies and Applications of Artificial Intelligence (TAAI 2012)*, November 2012.

[10] P. Comon. Independent component analysis, a new concept? *Signal Process*, 36(3):287–314, Apr. 1994.

[11] D. Cook, K. D. Feuz, and N. C. Krishnan. Transfer learning for activity recognition: A survey. *Knowledge and Information Systems*, 2012.

[12] T. M. Cover and J. A. Thomas. *Elements of Information Theory*. Wiley-Interscience, 1991.

[13] K. Crammer, M. Kearns, and J. Wortman. Learning from multiple sources. In B. Schölkopf, J. Platt, and T. Hoffman, editors, *Advances in Neural Information Processing Systems 19*, pages 321–328. MIT Press, Cambridge, MA, 2007.

[14] J. Edmonds. Paths, trees, and flowers. *Canadian Journal of Mathematics*, 17:449–467, 1965.

[15] D. M. Endres and J. E. Schindelin. A new metric for probability distributions. *IEEE Transactions on Information Theory*, 49(7):1858–1860, Sept. 2006.

[16] K. D. Forbus, D. Gentner, and K. Law. Mac/fac: A model of similarity-based retrieval. *Cognitive Science*, 19(2):141–205, 1995.

[17] A. Frank and A. Asuncion. UCI machine learning repository, 2010.

[18] D. Gale and L. S. Shapley. College admissions and the stability of marriage. *The American Mathematical Monthly*, 69(1):9–15, January 1962.

[19] D. Gentner. Structure-mapping: A theoretical framework for analogy. *Cognitive Science*, 7(2):155 – 170, 1983.

[20] A. Gretton, O. Bousquet, A. Smola, and B. Schölkopf. Measuring statistical dependence with hilbert-schmidt norms. In *Proceedings of the 16th international conference on Algorithmic Learning Theory*, ALT'05, pages 63–77, Berlin, Heidelberg, 2005. Springer-Verlag.

[21] P. Hall. On representatives of subsets. *Journal of the London Mathematical Society*, 10(1):26–30, 1935.

[22] D. H. Hu, V. W. Zheng, and Q. Yang. Cross-domain activity recognition via transfer learning. *Pervasive and Mobile Computing*, 7(3):344–358, June 2011.

[23] A. Hyvärinen, J. Karhunen, and E. Oja. *Independent Component Analysis*. John Wiley & Sons, 2001.

[24] I. T. Jolliffe. *Principal Component Analysis*. Springer Series in Statistics. Springer New York, second edition, 2002.

[25] E. Keogh. Data mining and machine learning in time series databases. Tutorial in SIGKDD 2004, 2004.

[26] E. Keogh, S. Lonardi, and C. A. Ratanamahatana. Towards parameter-free data mining. In *Proceedings of the tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 206–215, Seattle, WA, August 2004.

[27] M. Klenk and K. Forbus. Domain transfer via cross-domain analogy. *Cognitive Systems Research*, 10(3):240–250, Sept. 2009.

[28] S. Kullback and R. A. Leibler. On information and sufficiency. *The Annals of Mathematical Statistics*, 22(1):79–86, 1951.

[29] S.-I. Lee, V. Chatalbashev, D. Vickrey, and D. Koller. Learning a meta-level prior for feature relevance from multiple related tasks. In *Proceedings of the 24th international conference on Machine learning*, ICML '07, pages 489–496, New York, NY, USA, 2007. ACM.

[30] M. Li, X. Chen, X. Li, B. Ma, and P. M. Vitányi. The similarity metric. *IEEE Transactions on Information Theory*, 50(12):3250 – 3264, December 2004.

[31] J. Lin, R. Khade, and Y. Li. Rotation-invariant similarity in time series using bag-of-patterns representation. *Journal of Intelligent Information Systems*, 2012.

[32] M. M. H. Mahmud and S. Ray. Transfer learning using kolmogorov complexity: Basic theory and empirical evaluations. In J. C. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *Advances in Neural Information Processing Systems 20*, pages 985–992. MIT Press, Cambridge, MA, 2008.

[33] A. Martins. String kernels and similarity measures for information retrieval. Technical report, Institute for Systems and Robotics, CMU-Portugal, 2006.

[34] L. Mihalkova, T. Huynh, and R. J. Mooney. Mapping and revising markov logic networks for transfer learning. In *Proceedings of the 22nd national conference on Artificial intelligence - Volume 1*, AAAI'07, pages 608–614. AAAI Press, 2007.

[35] L. Mihalkova and R. J. Mooney. Transfer learning by mapping with minimal target data. In *Proceedings of the AAAI-08 Workshop on Transfer Learning For Complex Tasks*, Chicago, IL, July 2008.

[36] S. J. Pan, I. W. Tsang, J. T. Kwok, and Q. Yang. Domain adaptation via transfer component analysis. In *Proceedings of the 21st international jont conference on*

Artifical intelligence, IJCAI'09, pages 1187–1192, San Francisco, CA, USA, 2009. Morgan Kaufmann Publishers Inc.

[37] S. J. Pan and Q. Yang. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1345–1359, 2010.

[38] N. Quadrianto, A. Smola, T. Caetano, S. Vishwanathan, and J. Petterson. Multitask learning without label correspondences. In J. Lafferty, C. K. I. Williams, J. Shawe-Taylor, R. Zemel, and A. Culotta, editors, *Advances in Neural Information Processing Systems 23*, pages 1957–1965, 2010.

[39] L. R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. In *Proceedings of the IEEE*, pages 257–286, 1989.

[40] R. Raina, A. Battle, H. Lee, B. Packer, and A. Y. Ng. Self-taught learning: transfer learning from unlabeled data. In *Proceedings of the 24th international conference on Machine learning*, pages 759–766, New York, NY, USA, 2007. ACM.

[41] P. Rashidi and D. J. Cook. Activity recognition based on home to home transfer learning. In *Proceedings of the AAAI Plan, Activity, and Intent Recognition Workshop*, pages 45–52, 2010.

[42] C. E. Shannon. A mathematical theory of communication. *The Bell System Technical Journal*, 27(3):379–423, July 1948.

[43] P. C. Shields. Two divergence-rate counterexamples. *Journal of Theoretical Probability*, 6(3):521–545, 1993.

[44] E. M. Tapia. *Activity Recognition in the Home Setting Using Simple and Ubiquitous Sensors Emmanuel Munguia Tapia*. PhD thesis, Massachusetts Institute of Technology, 2003.

[45] E. M. Tapia, S. S. Intille, and K. Larson. Activity recognition in the home setting using simple and ubiquitous sensors. In *Proceedings of PERVASIVE 2004*, 2004.

[46] S. Thrun. Is learning the n-th thing any easier than learning the first? In D. Touretzky and M. Mozer, editors, *Advances in Neural Information Processing Systems (NIPS) 8*, pages 640–646, Cambridge, MA, 1996. MIT Press.

[47] L. Torrey and J. Shavlik. *Handbook of Research on Machine Learning Applications and Trends*. IGI Global, Aug. 2009.

[48] V. Van Asch. *Domain similarity measures: On the use of distance metrics in natural language processing*. Phd dissertation, University of Antwerp, Antwerp, Belgium, 2012.

[49] T. van Kasteren, G. Englebienne, and B. Kröse. Recognizing activities in multiple contexts using transfer learning. In *Proceedings of the AAAI Fall Symposium on AI in Eldercare: New Solutions to Old Problems*. AAAI Press, 2008.

[50] T. van Kasteren, G. Englebienne, and B. J. A. Kröse. Transferring knowledge of activity recognition across sensor networks. In *Pervasive Computing*, pages 283–300, 2010.

[51] H. Wang and Q. Yang. Transfer learning by structural analogy. In *Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence*, pages 513–518, 2011.

[52] B. Wei and C. Pal. Heterogeneous transfer learning with RBMs. In *Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence*, pages 531–536, 2011.

[53] Y. Yoshida, T. Hirao, T. Iwata, M. Nagata, and Y. Matsumoto. Transfer learning for multiple-domain sentiment analysis – identifying domain dependent/independent word polarity. In *Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence*, 2011.

[54] Y. Zhang and D.-Y. Yeung. Multi-task learning in heterogeneous feature spaces. In *Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence*, 2011.

[55] Y. Zhu, Y. Chen, Z. Lu, S. J. Pan, G.-R. Xue, Y. Yu, and Q. Yang. Heterogeneous transfer learning for image classification. In *Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence*, August 2011.