Heterogeneous Transfer Learning for Activity Recognition using Heuristic Search Techniques

Kyle D. Feuz
Weber State University
Ogden, UT USA
kylefeuz@weber.edu

Diane J. Cook
Washington State University
Pullman, WA, USA
cook@eecs.wsu.edu

Abstract

Purpose - Many pervasive computing applications require information about the activities currently being performed, but activity recognition algorithms typically require substantial amounts of labeled training data for each setting. One solution to this problem is to leverage transfer learning techniques to reuse available labeled data in new situations.

Design/methodology/approach - In this paper, we introduce three novel heterogeneous transfer learning techniques that reverse the typical transfer model and map the target feature space to the source feature space and apply them to activity recognition in a smart apartment. We evaluate the techniques on data from 18 different smart apartments located in an assisted-care facility and compare the results against several baselines.

Findings - The three transfer learning techniques are all able to outperform the baseline comparisons in several situations. Furthermore, the techniques are successfully used in an ensemble approach to achieve even higher levels of accuracy.

Originality/Value - The techniques in this paper represent a considerable step forward in heterogeneous transfer learning by removing the need to rely on instance-instance or feature-feature co-occurrence data.

Keywords: Heterogeneous Transfer Learning, Domain Adaption, Genetic Algorithms, Activity Recognition, Heuristic Search, Smart Environments

1 Introduction

Activity recognition is an important problem for many different applications including health monitoring, automatic security surveillance, and home/office automation. However, most activity recognition algorithms require significant amounts of labeled data which may not be readily available. Ideally we would like to be able to use labeled data from a different domain to improve learning in the target domain. One example would be to use labeled data from one or more smart apartments to recognize activities in a new smart apartment which may have a different layout, different residents, or different lifestyles or behavioral patterns. Another example would be using the labeled data from a smart apartment to perform activity recognition in a smart office.

Traditional supervised machine learning techniques rely on the assumptions that the training data and test data have similar probability distributions and that the classification task is the same for both datasets. However, in the previous examples the

source and target data are clearly drawn from different probability distributions. In these cases, traditional machine learning techniques often fail to correctly classify the test data.

Transfer learning techniques have been proposed to specifically handle these types of situations. Transfer learning algorithms seek to apply knowledge learned from a previous task to a new, but related, task. The intuition behind transfer learning stems from the ability of humans to extend what has been learned in one context to a new context. In the field of machine learning, the benefits of transfer learning are numerous; less time is spent learning new tasks, less information is required of experts (usually human), and more situations can be handled effectively, making the learned model more robust. These potential benefits have led researchers to apply transfer learning techniques to many domains with varying degrees of success.

Most transfer learning techniques focus on situations where the difference between the source and target domains stems mainly from differences in the marginal probability distributions of the domains or different task labels (Cook, et al., 2012; Pan & Yang, 2010). Daumé and Marcu model the probability distribution using a mixture model with shared and disjoint components (2006). Blitzer et al. (2006; 2007) propose Structural Correspondence Learning (SCL) to use the correlation between certain pivot features (which have the same semantic meaning in both domains) and other features to create a common feature representation.

Heterogeneous transfer learning focuses on transfer learning problems where the source and target domains are different because they have different feature spaces. Dai et al. attempt solving the heterogeneous transfer learning problem by extending the risk minimization framework (Lafferty & Zhai, 2001) and developing a translator between feature spaces based upon co-occurrence data (feature-feature, feature-instance, instance-feature, or instance-instance) between the source and target datasets (Dai, et al., 2008). Prettenhofer extends SCL to the heterogeneous transfer learning case by using a translation oracle (i.e. a domain expert or bi-lingual dictionary) to enumerate several pivot features. These pivot features are then correlated to the other features in both domains and a cross-lingual classifier is trained

(Prettenhofer & Stein, 2011).

Manual mapping strategies have also been used to overcome differences in the feature spaces. For example, Van Kasteren et al. (2008; 2010) group sensors by their location/function. Sensors in the source domain are then mapped to similar sensors in the target domain. Rashidi and Cook also map sensors based on location/function but apply additional transfer learning techniques to better align the source and target datasets (Rashidi & Cook, 2010; 2011). Our approach, Feature-Space Remapping (FSR), eliminates the need to manually map the feature spaces as this is handled by the algorithm. Additional domain adaptation approaches can then be applied to further improve the knowledge transfer. FSR requires the manual specification of meta-features but this specification only occurs once and can be applied to map multiple source and target domains. The techniques of both Rashidi and Van Kasteren require a mapping to be defined for each source and target pair. Additionally, the manual mapping strategies are domain dependent, while FSR is applicable to a variety of different problems.

Each of the above mentioned heterogeneous techniques requires some form of linkage (co-occurrence data, dictionaries, or domain experts) between the source and target dataset. FSR uses only a small amount of labeled data in the target domain to infer

relations to the source domain and can optionally operate without any labeled data in the target domain or other linkage data.

With heterogeneous learning, transfer between vastly different domains becomes feasible. The majority of heterogeneous transfer learning techniques map the source feature space to the target feature space or to map both the source and target feature space to a shared feature space. However, we show that by reversing this model and mapping the target feature space to the source feature space one can leverage an existing hypothesis in the source feature space to find a better mapping between feature spaces. Additionally, by mapping the target feature space to the source feature space one can easily create ensemble learners which further improve the accuracy of the proposed techniques.

In this paper we propose three novel heterogeneous transfer learning techniques: Feature-Space Remapping (FSR), Genetic Algorithm for Feature-Space Remapping (GAFSR), and Greedy Search for Feature-Space Remapping (GrFSR). All three techniques are capable of handling different feature spaces without the use of a translation oracle or instance-instance co-occurrence data. We term the technique a "remapping" because the original raw target data is already mapped onto a feature space and we remap the data to the source feature space. The FSR technique can be used in either the informed or uninformed transfer learning setting and we provide details for both cases. FSR uses only a small amount of labeled data in the target domain to infer relations to the source domain and can optionally operate without any labeled data in the target domain or other linkage data. GAFSR and GrFSR are both informed supervised transfer learning techniques requiring labeled data be available in both the source and target domain. For simplicity, we present the techniques here assuming the feature-space is a vector of real-valued numbers. However, it is straightforward to extend the approaches to handle categorical or discrete values as well.

We hypothesize that using FSR will allow for the successful transfer of knowledge between heterogeneous source and target domains without requiring the typical co-occurrence data or resorting to manually mapping the source and target domains to a shared domain. We test this hypothesis on data gathered from multiple smart apartments with varying layouts and sensor configurations and compare the performance to a manual mapped strategy and to situations in which no transfer is performed.

In addition to presenting FSR for transferring knowledge from a single source domain to a target domain, we also show how FSR can effectively combine the information from multiple source domains by using an ensemble learner to increase the classification accuracy in the target domain. We illustrate our techniques using examples from activity recognition.

2 Background and Problem Definition

Many of the ideas and principles of machine learning have originated from comparisons and analogies to human learning. The same is true with transfer learning. The ability to identify deep, subtle connections, what we term *transfer learning*, is the hallmark of human intelligence. Byrnes (1996) defines transfer learning as the ability to extend what has been learned in one context to new contexts. Thorndike and Woodworth (1901) first coined this term as they explored how individuals transfer learned concepts between

contexts that share common features. Barnett and Ceci provide a taxonomy of features that influence transfer learning in humans (2002).

In the field of machine learning, transfer learning is studied under a variety of names including learning to learn, life-long learning, knowledge transfer, inductive transfer, context-sensitive learning, and meta-learning (Arnold, et al., 2007; Elkan, 2001; Thrun & Pratt, 1998; Vilalta & Drissi, 2002). It is also closely related to self-taught learning, multi-task learning, domain adaptation, and co-variate shift. Because of this broad variance in the terminology used to describe transfer learning it is helpful to provide a formal definition of the terms we will use throughout the rest of this paper.

2.1 *Definitions*

Definitions for domain and task have been provided by Pan and Yang (2010):

Definition 1 (Domain) *A domain D is a two-tuple* $(\mathcal{X}, P(X))$*. $\mathcal{X}$ is the feature space of D and* $P(X)$ *is the marginal distribution where* $X = \{x_1,...,x_n\} \in \mathcal{X}$*.*

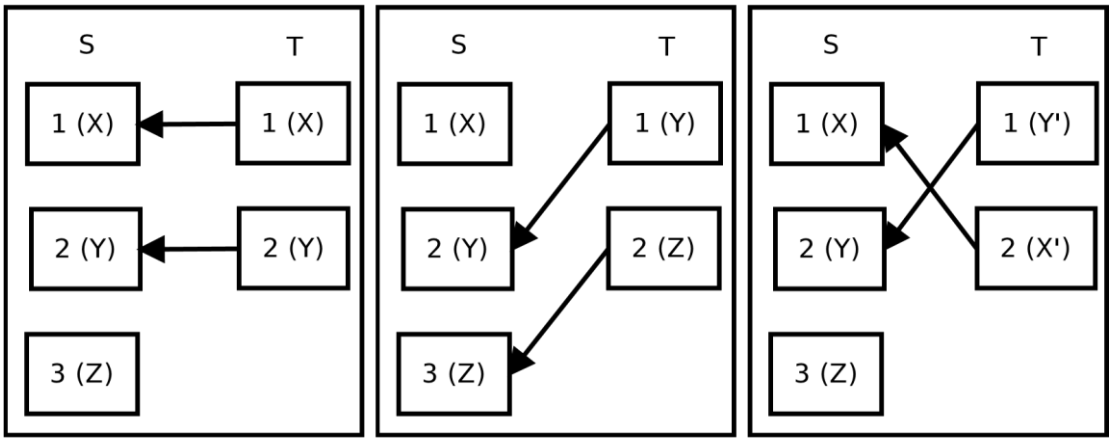
Definition 2 (Task) *A task T is a two-tuple* $(Y, f(\,))$ *for some given domain D. Y is the label space of D and* $f(\,)$ *is an objective predictive function for D.* $f(\,)$ *is sometimes written as a conditional probability distribution* $P(y|x)$*.* $f(\,)$ *is not given, but can be learned from the training data.*

Using these terms, we can now define transfer learning. In this paper we use the definition given by Cook et al. (2012) which is similar to that presented by Pan and Yang (Pan & Yang, 2010) but allows for transfer learning from multiple source domains.

Definition 3 (Transfer Learning) *Given a set of source domains* $DS = D_{s1},...,D_{sn}$ *where* $n > 0$*, a target domain,* D_t*, a set of source tasks* $TS = T_{s1},...T_{sn}$ *where* $T_{si} \in TS$ *corresponds with* $D_{si} \in DS$*, and a target task* T_t *which corresponds to* D_t*, transfer learning helps improve the learning of the target predictive function* $f_t(\,)$ *in* T_t *where* $D_t 6\in DS$ *and* $T_t 6\in TS$*.*

This definition of transfer learning is broad and encompasses a large number of different transfer learning scenarios. The source tasks can differ from the target task by having a different label space, a different predictive function for labels in that label space, or both. The source data can differ from the target data by having a different domain, a different task, or both. The FSR algorithm focuses on the challenge of the source and target domain coming from different feature spaces. This is commonly referred to as heterogeneous transfer learning in the literature and is formally defined below.

Definition 4 (Heterogeneous Transfer Learning) *Given a set of source domains* $DS = D_{s1},...,D_{sn}$ *where* $n > 0$*, a target domain,* D_t*, a set of source tasks* $TS = T_{s1},...T_{sn}$ *where* $T_{si} \in TS$ *corresponds with* $D_{si} \in DS$*, and a target task* T_t *which corresponds to* D_t*, transfer learning helps improve the learning of the target predictive function* $f_t()$ *in* T_t *where* $\mathcal{X}_t \cap (\mathcal{X}_{s1} \cup ...\mathcal{X}_{sn}) = \emptyset$*.*



(a) Map 1 (b) Map 2 (c) Map 3

Figure 1: Example mappings from target T (two-dimensional data) to source S (three-dimensional data)

Although FSR, GAFSR, and GrFSR focus on different feature spaces, they do not rely on the other dimensions of the transfer learning problem remaining constant. Indeed the datasets we use in the experimental section have differences in the marginal probability distributions as well as in the label space. As with all transfer learning problems we do rely on the basic assumption that there exists some relationship between the source and target areas which allows for the successful transfer of knowledge from the source to the target.

When the feature spaces of the domains are different, we assume that they can be different both in terms of the number of dimensions and in the organization of the dimensions. To illustrate this point, consider two different domains, one consisting of two dimensional data and the other consisting of three dimensional data. It could be the case that the first two dimensions are the same in both domains (see Figure 1a); however, it could also be the case that the first two dimensions of the target domain correspond with the last two dimensions of the source domain (see Figure 1b), or perhaps only the first dimension of the target domain corresponds with the last dimension of the source domain. It may even be the case that the dimensions are entirely different, but a mapping between dimensions could still allow the knowledge gained in one domain to be used effectively in the other domain (see Figure 1c). FSR learns a mapping from the target feature space to the source feature space regardless of the exact differences between dimensions.

In traditional machine learning, there are three basic types of techniques that are utilized based upon the availability of labeled data, supervised, unsupervised, and semi-supervised. In transfer learning, the availability of labeled data can be different in the source and target domains, thus four general classes of techniques arise, informed supervised, informed unsupervised, uninformed supervised, uninformed unsupervised. We follow the definitions of Cook et al. (2012), where informed or uninformed refers to the presence or absence, respectively, of labeled data in the target domain, and supervised or unsupervised refers to the presence or absence of labeled data in the source domain. A few works have explored unsupervised transfer learning techniques (Dai, et al., 2008; Wang, et al., 2008) but we focus on supervised transfer learning in this paper. GAFSR and GrFSR are developed as informed supervised learning techniques. FSR can be applied to either the informed or uninformed case but for consistency we focus on the informed supervised learning technique.

2.2 *Illustrative Example*

Before describing GAFSR, GrFSR, and FSR, we put forward an example transfer learning scenario to illustrate the concepts introduced throughout the discussion. To that end, let us consider the transfer learning problem for activity recognition in a smart environment using ambient sensors.

Ambient sensors are typically embedded in an individual's environment. Examples of ambient sensors may include motion detectors, door sensors, object vibration sensors, pressure sensors, and temperature sensors. As the name indicates, these sensors are designed to disappear into the environment while collecting a variety of activity related information such as human movements in the environment induced by activities, interactions with objects during the performance of an activity, and changes to illumination, pressure and temperature in the environment due to activities. Table 1 shows some example data from a smart home with ambient motion sensors.

Date	Time	Sensor	Value
2011-06-15	03:41:50.30088	M021	OFF
2011-06-15	03:41:50.402649	MA020	OFF
2011-06-15	03:44:50.862962	M021	ON
2011-06-15	03:44:51.929508	M021	OFF
2011-06-15	04:41:28.179357	M021	ON
2011-06-15	04:41:29.333803	M021	OFF
2011-06-15	05:33:44.024833	M021	ON
2011-06-15	05:33:45.118382	M021	OFF
2011-06-15	06:33:30.363675	M021	ON
2011-06-15	06:33:31.437863	M021	OFF
2011-06-15	06:33:33.878588	M021	ON
2011-06-15	06:33:35.956492	M021	OFF
2011-06-15	08:45:45.685723	M021	ON
2011-06-15	08:45:46.789252	M021	OFF
2011-06-15	08:46:03.646237	M021	ON
2011-06-15	08:46:03.817155	MA020	ON
2011-06-15	08:46:08.513192	M021	OFF
2011-06-15	08:46:08.712314	MA020	OFF
2011-06-15	08:46:09.87972	MA020	ON
2011-06-15	08:46:12.103082	MA020	OFF
2011-06-15	08:46:21.859339	MA020	ON
2011-06-15	08:46:22.752142	M021	ON
2011-06-15	08:46:23.885996	M021	OFF
2011-06-15	08:46:25.199775	MA020	OFF
2011-06-15	08:46:26.713111	MA020	ON
2011-06-15	08:46:27.590115	M019	ON
2011-06-15	08:46:29.876241	MA020	OFF
2011-06-15	08:46:30.760636	M019	OFF
2011-06-15	08:46:32.587806	M018	ON
2011-06-15	08:46:36.329587	MA013	ON
2011-06-15	08:46:37.117772	M018	OFF
2011-06-15	08:46:45.86861	MA013	OFF

Table 1: Sample of Sensor Events

Suppose there are two homes (a source home and a target home) equipped with these ambient sensors. The source home already has an activity recognition model trained for that home. The target home does not yet have an activity recognition model trained. In order to use the model from the source home to recognize activities in the target home, they must use a common feature-space. A common approach to activity recognition using ambient sensors is to formulate the problem as a bag of sensors approach over some sliding window of time or sensor events. This means that the sensors from one home must

be mapped onto the sensors from the other home. Specifically, the features of one domain must map onto the features (or dimensions) of the other domain. This could be accomplished by mapping the sensors in the target home to the sensors in the source home, mapping the sensors in the source home to sensors in the target home, or mapping both the source and target sensors to a common set of generic labels (for example, location-based mapping such as kitchen, bedroom, etc.).

This mapping is just the initial step in the transfer learning. Once a shared feature-space is achieved, additional transfer learning may be necessary to resolve differences in the marginal probabilities (the residents in one home may spend half the day sleeping, while the residents in the other home only sleep 6 hours a day) or differences in the classification task (the set of activities recognized may be different). The techniques we present here focus on achieving this initial transformation of the feature-space.

3 Methods

Traditionally, domain adaptation problems have focused on the case when $D_s \neq D_t$, usually because $P(X_s) \neq P(X_t)$. For example, in activity recognition, the behavior of an individual may change over time, multiple individual may utilize the same space differently. This creates situations where the feature-space has not changed but the probability distribution of the features over that feature-space has changed. When domain adaptation has been applied to problems where $\mathcal{X}_s \neq \mathcal{X}_t$ there is usually a trivial transformation between feature spaces. An example of this is found in document classification, where the domain dimensions are typically word counts in each document. To compare documents with different words, a user can set the word counts for the unseen words to zero. This allows the user to easily define a common feature space between documents. Additional transfer learning techniques may still be necessary because $P(X_s) \neq P(X_t)$ but the initial feature-space transformation is trivial. This trivial transformation works because the semantic meaning of the dimensions is assumed to be known.

Activity recognition aims to identify activities as they occur based on data collected by sensors. Advances in pervasive computing and sensor networks have resulted in the development of a wide variety of sensor modalities that are useful for gathering information about human activities. The feature-spaces in these type of activity recognition problems are often difficult to align because there is no trivial transformation between feature-spaces.

In this work we present a heterogeneous transfer learning algorithm where the feature space must be transformed in a non-trivial manner. The semantic meaning of the dimensions is assumed to be either unknown or incompatible between the source and target domains. In the activity recognition domain this is equivalent to having sensor values but not knowing from which sensor (type or location) it originated. Unlike many other heterogeneous transfer learning techniques, we do not rely on co-occurrence data such as dictionaries, social annotations of images, or multi-view data. Additionally, we do not assume that $P(Y_s|X_s) = P(Y_t|X_t)$ or even that $Y_s = Y_t$ but we do assume that they must still be related.

To achieve the desired feature-space transformation, we view the problem as a new machine learning task to learn a mapping from each dimension in the target feature space to a corresponding dimension in the source feature space. More formally this can be

written as follows: Given source data X_s, target data X_t and a hypothesis $H_s : \chi_s \rightarrow Y_s$ find a mapping $\theta(\chi_t, \chi_s)$ such that $error_\theta(H_s)$ is minimized where $error_\theta(H_s)$ represents the empirical error on the target domain by using H_s on the mapped target data. Notice the distinction between this problem definition and other approaches typically applied to heterogeneous transfer learning. Traditional heterogeneous transfer learning approaches usually map source features to target features or source and target features to a common feature space and then learn a hypothesis on this common feature space. In our approach however, we map the target features to source features and we use an already learned hypothesis to guide the mapping process and avoid the duplication of work. If the mapping process proceeded in the other direction we would need to relearn a new hypothesis for each step of the search which would greatly increase the computational complexity of these techniques. By mapping from target to source we also gain the ability to combine multiple data sources through ensemble learning which will be discussed in Section 4. It is possible to relearn a new hypothesis after performing the mapping. It is also possible to apply additional transfer learning approaches after first obtaining a unified feature-space.

The number of possible mappings between source and target feature spaces grows exponentially as the number of features increases. Even for lower dimensional data, searching through all possible mappings quickly becomes computationally infeasible. First we present Genetic Algorithms for Feature-Space Remapping (GAFSR) which explores the search space using random permutations of possible mappings. This method is the most computationally expensive but also explores the largest amount of mapping space. Next we present Greedy Search for Feature-Space Remapping (GrFSR) which applies the fitness function of the genetic algorithm to greedily select an approximation to the optimal mapping without searching through all possible mappings. Finally, we present Feature-Space Remapping (FSR) which uses less computationally expensive heuristics to select an approximation to the optimal mapping.

The techniques we propose generate a many-to-one mapping. This is because multiple dimensions (features) in the target space can be mapped to a single feature in the source space but one feature in the target domain will never map to multiple features in the source domain. We could make the mapping stricter by enforcing a one-to-one mapping (with null mappings allowed) or we could relax the mapping by allowing weighted many-to-many mappings. However, if we allowed a many-to-many mapping the search space (which is already too large for brute-force searching) would grow even larger. For many situations, a many-to-one mapping makes the most sense intuitively. For example, consider a hallway which is lined with several narrow-view motion sensors in one apartment and a hallway which has a single wide-view motion sensor in another apartment. Each narrow-view motion sensor could map to the single wide-view motion sensor in the other apartment but the wide-view motion sensor should just map to the single narrow-view motion sensor which best encapsulates its behavior.

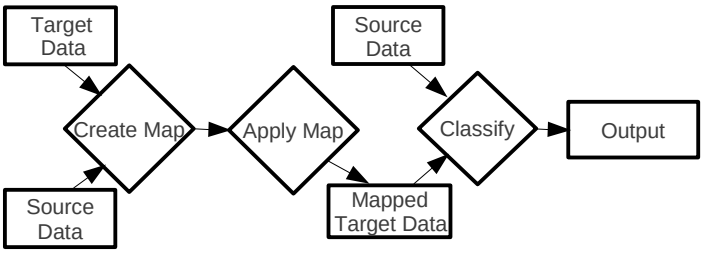


Figure 2: Flowchart of the mapping process

After the mapping has been obtained, the mapping must be applied to the target data to be classified using the hypothesis learned on the source data. Because these techniques produces a many-to-one mapping, the procedure for combining the multiple dimensions must also be defined. For dimensions with numerical values, one could use an aggregate value such as minimum, maximum, total, or average. For categorical values, one could use a voting protocol. For each instance in the target data the features are mapped to the source features. When multiple features in the target data are mapped to single feature in the source data, the feature values are combined using the specified aggregation protocol. In this work we use the summed value to aggregate target features mapped to the same source feature. The entire process is summarized in Figure 2. The three techniques differ in how they create the map but the rest of the steps are all identical.

3.1 *Genetic Algorithm Feature-Space Remapping*

The goal of the GAFSR technique is to find a near optimal mapping $\theta(\chi_t, \chi_s)$ such that the error of the hypothesis on the target data is minimized. If n is the number of features in χ_s and m is the number of feature in χ_t then there are n^m possible mappings, making it impractical to try all possible mappings. Instead, GAFSR uses a standard genetic algorithm approach to explore the search space looking for reasonable solutions.

Genetic algorithms are a class of local search techniques which has been studied for the past several decades (Mitchell, 1998; Russell & Norvig, 2010). The motivation for genetic algorithms is rooted in the biological process of reproduction and evolution (Goldberg & Holland, 1988). The basic idea is to start with a random population of strings (chromosomes) and evaluate their fitness according to a specified function (the fitness function). Chromosomes pairs are then selected (usually probabilistically according to the normalized fitness score) and the selected chromosomes are mated via crossover to produce new offspring. This process is then repeated a number of times until a stopping criterion is met. Pseudo-code for GAFSR is found in Algorithm 1. The key components of a genetic algorithm are: the chromosome definition, the fitness function, and the mutation parameters. Each are described below.

The chromosome is defined such that each sensor in the target dataset is a gene with n+1 possible values (1 for each source feature plus a null feature), thus the chromosome is composed of m genes with n + 1 possible values for each gene. From a practical standpoint, n can be seen as the number of sensor in the source domain and m can be seen as the number of sensor in the target domain when using a bag of sensors approach to the activity recognition problem.

Algorithm 1: GAFSR Algorithm

Data: X_t Target Features
Data: X_y Source Features
Data: s population size, g number of generations
Data: c cross-over rate, r mutation rate
Generate s initial random mappings from X_t to X_y // i.e. Chromosomes
for $i \leftarrow 0$ **to** g **do**
    Evaluate fitness of each mapping;
    Select pairs of mappings probabilistically weighted according to fitness;
    With probability c, Swap portions of mappings between the selected pair;
    With probability r, Mutate the mappings;
Evaluate fitness of each mapping;
Return mapping with best fitness;

We compare two different fitness functions based upon the unweighted average recall (UAR) and the accuracy (ACC) of the target dataset obtained using a naïve Bayes classifier which has been trained on the source dataset. The unweighted average recall is given by Equation 1 and the accuracy is given by Equation 2. In both of these equations N is the total number of instances, K is the number of labels, and A is the confusion matrix where A_{ij} is the number of instances of class i classified as class j.

$$UAR = \frac{1}{K}\sum_{i=1}^{K}\frac{A_{ii}}{\sum_{j=1}^{K}A_{ij}} \tag{1}$$

$$ACC = \frac{1}{N}\sum_{i=1}^{K}A_{ii} \tag{2}$$

The first fitness function, given in Equation 3, is defined as just the UAR of the target dataset obtained using a naïve Bayes classifier which has been trained on the source dataset. We use the average recall instead of the overall accuracy because the datasets are imbalanced. A large percentage of the instances are represented by only a few class labels. Using the unweighted average class recall is one technique for accounting for this imbalance (van Kasteren, et al., 2008). The second fitness function, given in Equation 4, is defined as the twice the UAR plus the overall accuracy (ACC) on the target dataset obtained using a naïve Bayes classifier which has been trained on the source dataset. This function is chosen to improve the overall accuracy obtained by the mapping technique while still preserving the high unweighted average recall.

$$F1 = UAR \tag{3}$$

$$F2 = ACC + 2 * UAR \quad (4)$$

The parameters of the genetic algorithm are chosen using limited validation testing to find parameters which yield decent results. They are set to the following values:

- Population Size: 118
- Mutation rate: .06
- Crossover rate: .80
- Crossover type: 2-point cross-over
- Number of Generations: 100

In addition, we use the technique referred to as elitism where the best solution so far is preserved across generations. This prevents the algorithm from losing the best solution due to the random mutations and crossovers.

The asymptotic runtime of the proposed genetic algorithm is $O(S * G * N * d_s)$ where S is the size of the population, G is the number of generations N is the number of labeled target instances and d_s is the number of dimensions in the source feature-space. We have purposely excluded the cost of creating the population for each generation because the cost of the fitness function shown here is the dominating factor. For the size of the activity recognition datasets we test here, $S * G \approx d^2_s$ giving us an asymptotic runtime of $O(N * d_s^3)$.

3.2 *Greedy Search for Feature-Space Remapping*

An alternative to genetic methods for searching a space is applying a greedy search, which does not rely on the partially-random biologically-inspired search mechanism found in genetic algorithms (Russell & Norvig, 2010). To compare our genetic solution to a greedy approach, we introduce GrFSR which applies the same fitness function employed by the genetic algorithm to greedily search through the mapping space and find an approximation to the best mapping function.

The greedy algorithm selects a single feature in the target domain to consider. It then maps this feature to all possible features in the source domain one at a time (including the null feature, which in effect ignores the corresponding feature in the target domain) while all other features in the target domain are mapped to null. The resulting mapping is applied to the labeled target data and tested using the hypothesis obtained from the source data. The mapping that produces the best result according to Equation 4 is selected as the best mapping for that target feature. This is repeated for all the target features. A final mapping is produced by combining the best mapping produced for each target feature. Pseudo-code for the algorithm is given in Algorithm 2.

The asymptotic runtime of GrFSR is $O(d_s * d_t * N * d_s)$ where d_s is the number of dimension in the source feature-space, d_t is the number of dimension in the target feature-space and N is the number of labeled data instance in the target domain. This runtime is equivalent to $O(N * d^3)$ if $d_s \approx d_t$.

Algorithm 2: GrFSR Algorithm

Data: X_t Target Features
Data: X_s Source Features
for $x \in X_t$ **do**
  for $y \in X_s$ **do**
    fit[y] ← Fitness($x \rightarrow y$);
  mapping[x] ← max(fit);
Return mapping;

3.3 *Feature Space Remapping*

In Feature Space Remapping, rather than exploring the entire search space of possible mappings we instead use heuristics to select a mapping that approximate the optimal mapping. Feature-feature, feature-instance or instance-instance co-occurrence data could be used to guide the search but FSR operates under the assumption that this type of data is not available. Instead FSR computes meta-features as a means to relate source and target features. These meta-features can be defined and computed multiple ways which will be discussed in Section 3.4. Algorithm 3 shows the pseudo-code for the FSR technique and each step is discussed in detail below. To simplify the presentation of the FSR algorithm, for now let us assume that meta-features have already been calculated for the source and target features. One can think of the meta-features as a vector of numbers assigned to each feature in the source and target space. These vectors can then be compared to each other to find features with similar meta-features.

FSR computes a similarity matrix S between source features and target features. This is done by computing a similarity score for each feature-feature pair based upon the meta-features computed for the given features. The similarity score is computed as the average similarity between the source and target meta-feature values. Formally, this score is given by Equations 5 and 6.

$$S_{xy} = \frac{1}{N}\sum_{i=1}^{N} \Omega\left(m_x^i, m_y^i\right) \tag{5}$$

where x is the *xth* source feature, y is the *yth* target feature, N is the number of meta-features and Ω is the normalized similarity between two meta-features m_x^i and m_y^i, the *ith* meta-feature of feature x and y respectively. We calculate the normalized similarity between two meta-features as the absolute value of the difference between meta-feature values divided by the maximum possible difference between the meta-features to obtain a normalized value between 0 and 1. This is shown in Equation 6.

$$\Omega\left(m_x^i, m_y^i\right) = 1 - \frac{\left|m_x^i - m_y^i\right|}{\max(m_x^i, m_y^i \forall x \in D_s \forall y \in D_t) - \min(m_x^i, m_y^i \forall x \in D_s \forall y \in D_t)} \tag{6}$$

Algorithm 3: FSR Algorithm

Data: X_t Target Features
Data: X_y Source Features
for $x \in X_t$ **do**
  metas[x] ← ComputeMetaFeatures(x);
for $x \in X_s$ **do**
  metas[x] ← ComputeMetaFeatures(x);
for $x \in X_t$ **do**
  for $x \in X_s$ **do**
    similarity[x][y] ← ComputeSimilarity(metas[x], metas[y])
for $x \in X_t$ **do**
  mapping[x] ← max(similarity[x]);
Return mapping;

If the meta-feature values are all positive, which is the case for the experiments we show here, the normalized similarity equation can be simplified to:

$$\Omega\left(m_x^i, m_y^i\right) = 1 - \frac{\left|m_x^i - m_y^i\right|}{\max(m_x^i, m_y^i \forall x \in D_s \forall y \in D_t)} \tag{7}$$

FSR computes a mapping L : y → x by selecting source feature x with maximal similarity to target feature y as given by the similarity matrix S.

$$\mathrm{L(y)} = \underset{x \in D_S}{\operatorname{argmax}}(S_{xy}) \tag{8}$$

If we assume the meta-feature computation is linear, FSR has a running time of O($d_s * d_t + n + m$) where d_s and d_t is the dimensionality of the source and target data, respectively, and n and m are the number of source and target instances, respectively. This runtime is explained by the following observations. First, each dimension in the target domain is compared to each dimension in the source domain, resulting in the $d_s * d_t$ term. Second, assuming the meta-feature computation is linear in the number of data instances, then computing the meta-features requires $O(n + m)$ time. Finally, applying the mapping requires a single pass through the target data or $O(m)$ time.

As mentioned earlier, the defining and calculating of meta-features can be done in multiple ways. If some labeled target data is available, it can be used to calculate domain-independent meta-features (i.e. meta-features that can be applied to any heterogeneous transfer learning problem). We refer to this as Informed Feature Space Remapping (IFSR) because it requires the labeled target data. If no labeled target data is available then domain-dependent meta-features must be defined. We refer to this as Uninformed

Feature Space Remapping (UFSR) because it does not require the label target data. In this paper we only present the IFSR technique as it achieves better results than UFSR and is consistent with the other techniques.

3.4 *Informed Feature Space Remapping*

Searching through all possible mappings to find the mapping which minimizes the error of the hypothesis on the target data is computationally expensive. However, since the hypothesis has been learned using the source training data one would expect the error to be minimized by selecting mappings for which the feature-label co-occurrence data is similar in the source and target datasets. This leads to our first heuristic for mapping source and target features. IFSR computes the feature-label co-occurrence data for each feature in the source and target space by calculating the expected value of the feature given the label using the labeled training data. More formally, if $Y = Y_S \cup Y_T$ then the feature-label co-occurrence data for each feature and label is computed as:

$$E(x|c) = \frac{1}{n_c} \sum_{i=1}^{n} x_i \tag{9}$$

where x is the feature, c is the label such that $c \in Y$, n_c is the number of data instances with label c, x_i is the value of feature x on the *ith* data instance with a label of c. This assumes a real-valued number space. One could easily extend this to categorical values by using the count of occurrences of each category as an estimation of the probability that the given feature will have the given categorical value.

Each feature-label co-occurrence value now becomes a meta-feature for the given feature. Thus $E(x|c)$ is a meta-feature for feature x and x will have $z = |Y|$ such meta-features, one for each label c. Using feature-label co-occurrence data as a meta-feature keeps the FSR asymptotic run time within the previously stated bound of $O(d_S * d_T + n + m)$. This is because the meta-feature calculation is linear in the number of instances. We compute $E(x|c)$ for each label $c \in Y$. This can be done in a single pass through the datasets and thus requires $O(n + m + y)$ time. Typically $n >> y$ and $m >> y$ so this term can be simplified to just $O(n + m)$.

Additionally, using feature-label co-occurrence data for the meta-features provides domain independent meta-features so that meta-features for the specific problem do not need to be specified by a domain expert. Thus any domain for which labeled data exists can apply this feature mapping technique without setting any parameters, defining any relations, or defining any additional meta-features.

To understand why using the feature-label co-occurrence data as a heuristic to find an approximation to the optimal mapping works we go back to the original problem definition. Given source data X_S, target data X_T and a hypothesis $H_S : X_S \rightarrow Y_S$ find a mapping $\theta(X_T, X_S)$ such that $error_\theta(H_S)$ is minimized. This error is minimized by maximizing the number of agreements between $H_S(\theta(q))$ and $f_T(q)$ as shown in Equation 10 where q is a data instance in X_T and $\theta(q)$ is the mapped data in the source domain space.

$$\max_{\theta} \sum_{q \in X_t} \begin{cases} 1, & if\ H_s(\theta(q)) = f_t(q). \\ 0, & if\ H_s(\theta(q)) \neq f_t(q). \end{cases} \tag{10}$$

A naïve Bayes classifier can learn a hypothesis by estimating $P(c)$ and $P(q_i|c)$ based upon their observed frequencies and applying Bayes rule to estimate the posterior probability $P(c|q)$. The class c with the highest posterior probability is selected as the class label for q (Mitchell, 1997). Thus, if the hypothesis is expressed as a naïve Bayes classifier and if we approximate the true predictive function $f_t()$ also using a naïve Bayes formulation then Equation 10 can be expressed as shown in Equation 11.

$$\max_{\theta} \sum_{q \in X_t} \begin{cases} 1, & if\ \max_{c \in Y} P(c) \prod_{i=1}^{d_t} P(\theta(q_i)|c) = \max_{c \in Y} P(c) \prod_{i=1}^{d_t} P(q_i|c). \\ 0, & if\ \max_{c \in Y} P(c) \prod_{i=1}^{d_t} P(\theta(q_i)|c) \neq \max_{c \in Y} P(c) \prod_{i=1}^{d_t} P(q_i|c). \end{cases} \tag{11}$$

Under this representation, selecting the mapping for each feature that has the most similar feature-label co-occurrence value can be seen as a greedy approximation to minimize the empirical error on the mapped target data. Indeed, when the feature values are restricted to either 0 or 1, the feature-label co-occurrence value $E(x|c)$ is equivalent to the estimation of the probability that the feature has a value of 1 given the class label, $P(x = 1|c)$.

4 Combining Multiple Data-sources

One of the major benefits of the above mapping approaches is that they can be used to combine data from multiple source domains in a straightforward manner. One example where multiple source domains might arise is a single individual with labeled data in multiple smart environments (home, office, car, etc.). Another example would be multiple smart apartments with labeled data which can be used to recognize activities in new smart apartment. An ensemble classifier can be built by mapping the target domain to each source domain and training a separate base classifier for each source domain. The output from these source classifiers can then be combined by the ensemble meta-classifier to make the final prediction. We refer to this as Ensemble Learning via Feature-Space Remapping (ELFSR).

Ensemble methods have been used in a variety of situations with great success. According to Hansen and Salamon, a necessary and sufficient condition for ensemble classifiers to be more accurate than any of the individual classifiers are for the classifiers to be accurate and diverse (Hansen & Salamon, 1990). An *accurate* classifier is one which has a classification accuracy better than random guessing (Dietterich, 2000). Two classifiers are diverse if the errors they make are different (and preferably uncorrelated) (Dietterich, 2000). Most ensemble techniques defined to date generate a set of diverse classifiers. Bagging, for example, generates classifiers by repeatedly subsampling the original data with replacement (Breiman, 1996). Boosting iteratively reweights samples based on the accuracy of the previous iteration (Goldberg & Holland, 1988). In ELFSR,

each classifier is drawn from a different domain, leading to a naturally diverse set of classifiers.

Once the classifiers are generated, the output must be combined to obtain the final result. Several approaches have been used including majority voting, weighted voting, summing the probabilities, and training a new learner on the output of the classifiers or *stacking* (Wolpert, 1992). Stacking is a supervised technique and thus requires additional labeled data to train the ensemble classifier. This means that stacking can be readily combined with IFSR, which already uses labeled data.

Work on ensemble classifiers for transfer learning has mainly focused on boosting techniques (Pan, et al., 2012; Xian-ming & Shao-zi, 2009; Yao & Doretto, 2010). As there has been very little work on transfer learning using voting or stacking ensemble classifiers, we compare the results of several different ensemble configurations using activity recognition from multiple smart apartments as the source domains and activity recognition for a different smart apartment as the target domain. Specifically, we consider two voting ensembles (a majority voting ensemble and a summation voting ensemble), and two stacking ensembles (via naïve Bayes and via a decision tree). The voting ensembles have the advantage of not requiring any labeled data in the target domain, while the stacking techniques require a small amount of labeled data.

4.1 *Voting Ensemble*

One of the simplest methods for combining multiple classifiers is through majority voting. Each classifier votes for the class label it predicts for the given instance and the label receiving the most votes wins.

The drawback to the majority voting ensemble classifier is that the ensemble throws away important information by only considering the most likely label as predicted by each classifier. The summation voting ensemble classifier rectifies this weakness by summing up the predicted probability of each label for each classifier and then assigning the label with the highest summed probability.

4.2 *Stacking*

In stacking, the output of each source classifier is fed into the ensemble classifier which then produces the final classification. Here we consider two different classification algorithms for the ensemble classifier, naïve Bayes and decision trees. One of the drawbacks to using stacking is the requirement of labeled data to train the ensemble classifier. Rather than test both FSR and IFSR with the stacking technique we only consider the result of using IFSR since IFSR already uses a small amount of labeled data in the target domain. We use stacking with IFSR without requiring any additional labeled data in the target domain.

5 Experimental Results

GAFSR, GrFSR, and FSR can be applied to a variety of different transfer learning problems. Here we evaluate the performance of these techniques in the activity recognition domain under the following scenarios:

- Experiment 1 (Fitness Function): Show the effect of the choice of the fitness function on the performance of GAFSR.

- Experiment 2 (FSR Comparison): Evaluate the performance of the FSR techniques in comparison to several baselines including a manual mapping strategy and a strategy that does not employ transfer learning.

- Experiment 3 (Learning Curve): Show the effect of the amount of labeled data available in the target domain on the performance of the FSR algorithm.

- Experiment 4 (ELFSR Comparison): Evaluate the performance of the ELFSR techniques in comparisons to several baseline techniques.

- Experiment 5 (ELFSR Learning Curve): Show the effect of the number of source apartments used in the ensemble techniques.

5.1 *Data*

We use a dataset consisting of data from 18 different smart apartments. The apartments are single residence assisted-living care facilities. Specific statistics for each apartment are found in Table 2. Each apartment is equipped with motion sensors and door sensors. The number of sensors range from 17 to 39 with an average of 28.7 sensors and a standard deviation of 6.21. The layout for the apartments is shown in Figure 3. Each dataset has been annotated with 37 different activities, shown in Table 3, with the total amount of labeled data spanning one month of time per dataset. Not all apartments have all 37 activity labels as indicated in the table. We consider all possible combinations of source and target datasets, yielding a total of 306 possible pairings. We use a single day of labeled data for the target domain and all 30 days of labeled data for the source domain.

Id	# Features	# Labels	# Instances	# UFSR Meta-Features	# IFSR Meta-Features
1	35	29	133157	1575	1295
2	17	26	53669	765	629
3	37	31	178137	1665	1369
4	29	29	57918	1305	1073
5	39	32	141181	1755	1443
6	26	32	149391	1170	962
7	26	30	183945	1170	962
8	26	28	98768	1170	962
9	34	30	102466	1530	1258
10	24	30	143145	1080	888
11	38	30	157736	1710	1406
12	24	29	135451	1080	888
13	32	32	116641	1440	1184
14	26	31	195611	1170	962
15	23	29	100255	1035	851
16	33	32	179693	1485	1221
17	23	29	92740	1035	851
18	24	30	117067	1080	888

Table 2: Summary statistics of the activity recognition dataset

We formulate the learning problem as that of mapping a sequence consisting of the most recent sensor events within a sliding window of length k to a label representing the activity to the last (most recent) event in the sequence. The sensor events preceding the last event define the context for this last event. Data collected in a smart home consists of events generated by the sensors. These are stored as a 4-tuple: (Date, Time, Sensor Id, Message) as shown in Table 1.

To perform activity recognition, we extract features from data point i, where the data point corresponds to a sensor event sequence of length k. The vector x_i includes values for the features summarized in Table 4. Each y_i corresponds to the activity label that is associated with the last sensor event in the sequence. A collection of data points, x_i, and the corresponding labels, y_i, are fed as training data to a classifier to learn the activity models in a discriminative manner. The classifier thus learns a mapping from the sensor event sequence to the corresponding activity label.

Activity	**Frequency**	**Activity**	**Frequency**
Enter Home	0.0031	Personal Hygiene	0.0545
Eat Lunch	0.0070	Leave Home	0.0026
Cook Dinner	0.0534	Eat Dinner	0.0100
Exercise	0.0002	Cook Lunch	0.0274
Wash Dinner Dishes	0.0127	Relax	0.0191
Read	0.0103	Wash Lunch Dishes	0.0077
Phone	0.0029	Evening Meds	0.0037
Eat Breakfast	0.0101	Watch TV	0.0405
Cook	0.0348	Wash Breakfast Dishes	0.0126
Eat	0.0066	Groom	0.0087
Housekeeping	0.0113	Toilet	0.0434
Wash Dishes	0.0088	Work At Desk	0.0004
Sleep Out Of Bed	0.0034	Work At Table	0.0253
Morning Meds	0.0053	Cook Breakfast	0.0320
Take Medicine	0.0036	Bed Toilet Transition	0.0156
Bathe	0.0175	Work	0.0329
Other Activity	0.2789	Entertain Guests	0.0837
Sleep	0.0407	Work On Computer	0.0498
Dress	0.0194		

Table 3: List of activities and the relative frequency of occurrence of each activity

5.2 *Fitness Function*

First, we consider the effect of the choice of fitness function on overall performance. Performance is measured using both the accuracy (given by Equation 2) and the unweighted average recall (given by Equation 1). We report both the accuracy and the recall because accuracy scores are biased towards the majority class. For balanced class distributions this has little effect on the metric, but it may not be suitable for unbalanced class distributions. Using the unweighted average recall eliminates this bias and treats all classes equally (van Kasteren, et al., 2008). Note that accuracy can also be considered as the average recall weighted by the number of instances in the class.

Figure 3: Apartment layouts for the 18 smart apartments used in the experiments.

Feature #	Value
1	Time of day of the latest sensor event in the sliding window
2	Day of week of the latest sensor event in the sliding window
3 to n + 3	Number of occurrences of each sensor in within the current window (n sensors)

Table 4: The feature vector describing a data point under the first feature representation.

5.3 *Fitness Function*

First, we consider the effect of the choice of fitness function on overall performance. Performance is measured using both the accuracy (given by Equation 2) and the unweighted average recall (given by Equation 1). We report both the accuracy and the recall because accuracy scores are biased towards the majority class. For balanced class distributions this has little effect on the metric, but it may not be suitable for unbalanced class distributions. Using the unweighted average recall eliminates this bias and treats all classes equally (van Kasteren, et al., 2008). Note that accuracy can also be considered as the average recall weighted by the number of instances in the class.

Figure 4 shows the results averaged over all 306 pairings. In this case we use the full 30 days of labeled data in both the source and target domain as we are interested only in the relative performance difference between the two fitness functions. As can be seen in the figure, including the overall accuracy in the fitness function significantly improves the accuracy without a significant drop in the unweighted recall.

5.4 *FSR Comparison*

Next, we compare the three proposed techniques, GAFSR, GrFSR, and IFSR, against several other baselines. GAFSR and GrFSR use the fitness function specified in Equation 4. IFSR uses the feature-label co-occurrence meta-features as described in Equation 9.

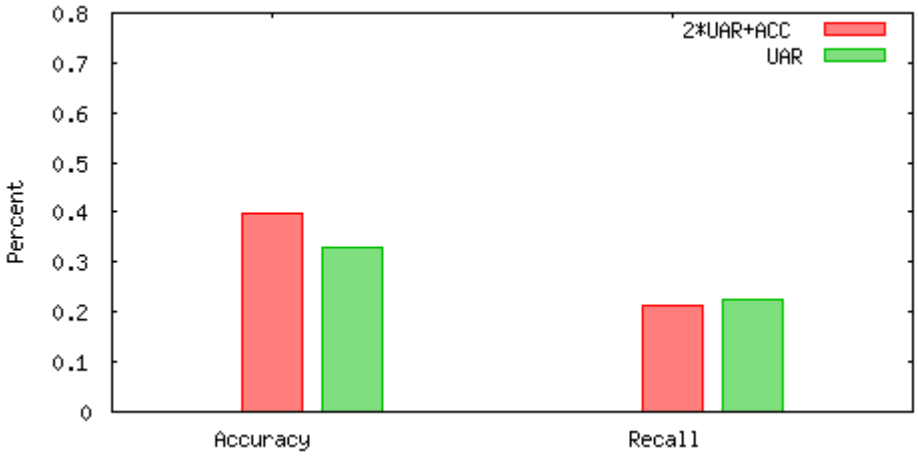


Figure 4: Average accuracy and recall scores over all 306 source-target pairings. Including the accuracy score in the fitness function improves accuracy without degrading the average recall.

The first baseline, *Manual*, uses the generalized sensor locations (kitchen, bedroom, etc.) to map sensors from one apartment to another. The second baseline, *None,* treats all sensor events as coming from a single source. Essentially this eliminates the sensor dimension and only considers the time of day and day of week of the activity. The *Manual* technique is the mapping technique currently used by most researchers in activity recognition (Cook, et al., 2012; Rashidi & Cook, 2011; van Kasteren, et al., 2008). It does not require any labeled data in the target domain, but it does require the manual definition of sensor locations. On the other hand, *None* provides a lower bound on the expected performance. The last baseline we consider, *Self* is a classifier trained and tested in the target domain. All of the techniques use a naïve Bayes classifier trained on the source domain and tested on the target domain. We considered other base classification algorithms such as SVMs, Decision Trees and Nearest Neighbors. However, since the meta-features used in IFSR are specifically related to naïve Bayes classification we have found that it gives good results without the computational overhead of some of the other methods. For comparison purposes, we also include results for IFSR when a decision tree has been used as the base classification method.

The results are shown in Figure 5. A one-way ANOVA is performed and the resulting p-value is less than .0001. The 95% confidence interval is depicted with the error bars. All three techniques match or beat the two baselines of *Manual* and *None*. As the amount of time spent exploring or computing a good mapping between the target and source domains increases the resulting accuracy, recall and precision scores also increases. GAFSR achieves the best performance scores under all three metrics but it also requires the most time to run, while IFSR uses the fewest number of computations but also has lower performance scores. Note that the performance gap between IFSR and GrFSR is much smaller than the gap between GAFSR and GrFSR and is even reversed for the precision metric. None of the techniques are able to match a classifier trained and tested in the source domain. This provides evidence that applying additional domain adaptation techniques may be beneficial.

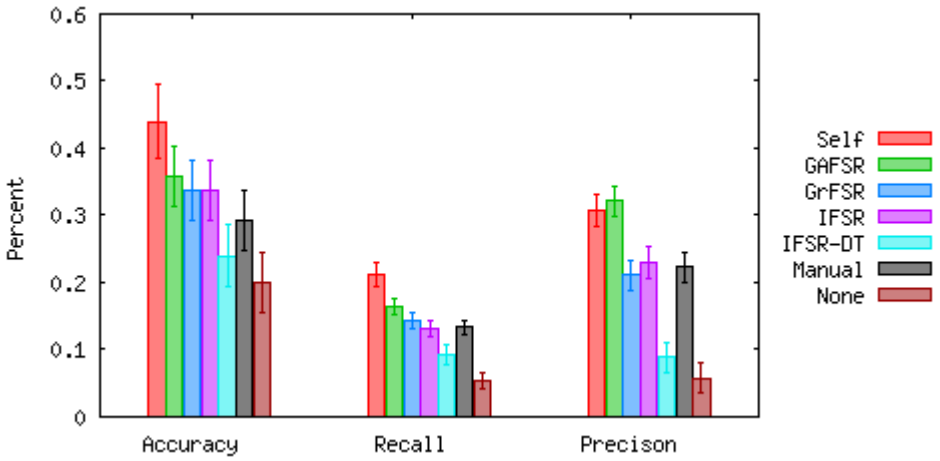


Figure 5: Classification accuracy and recall on the target domain using a single source domain. Manual and None provide baseline comparisons. Manual is the mapping specified by a domain expert. None does not apply any mapping at all. GAFSR, GrFSR and IFSR are all able to perform as good as or better than the Manual technique. The performance of GAFSR, GrFSR, and IFSR is ordered by the computational complexity of each technique, highlighting the benefit of exploring the mapping space at the cost of increased running times.

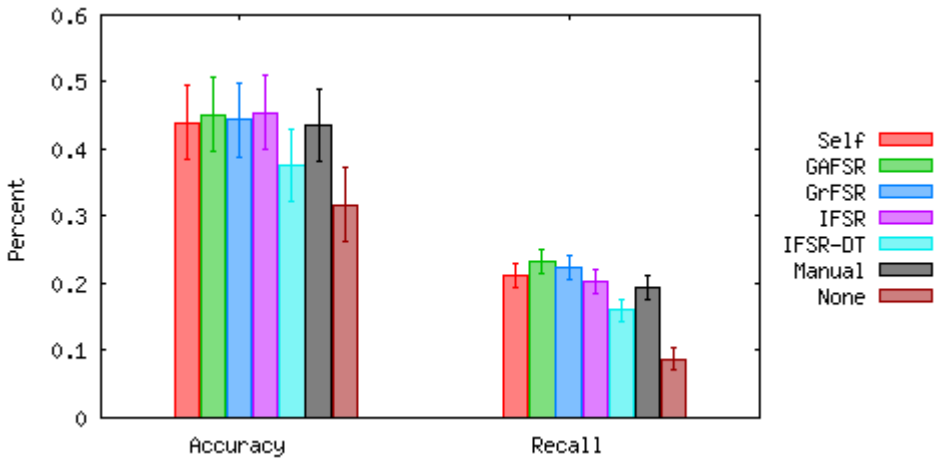


Figure 6: Classification accuracy and recall on the target domain using the best single source domain. This assumes that the best dataset to transfer from could be identified a priori. Manual and None provide baseline comparisons. Manual is the mapping specified by a domain expert. None does not apply any mapping at all. A one-way ANOVA is performed and the resulting p-value is less than .0005. The 95% confidence interval is depicted with the error bars.

The previously-discussed results are the average of 306 different mappings. Individual results show both higher and lower performance. One direction of transfer learning research focuses on how to select the best source dataset. Assuming this problem is solved then we could select the "best" source dataset for each target dataset. We do not claim that this contributes to avoid negative transfer, only that if negative transfer can be predicted and avoided we can improve the results. Figure 6 shows the results of using the best source dataset with the same mapping techniques discussed earlier. Under this scenario, the accuracy scores of the three techniques are nearly equivalent with *IFSR* actually performing the best. The recall scores of the three techniques continue to be ordered by the computational complexity of the technique. Again all three techniques are able to outperform the baseline techniques of *Manual* and *None* but this time they even match or beat the performance of *Self*. A one-way ANOVA is performed and the resulting p-value is less than .0005. The 95% confidence interval is depicted with the error bars.

5.5 *FSR Learning Curve*

The next experiment shows the effect of the amount of labeled target data on the accuracy and recall score of the *IFSR* algorithm. As in the previous experiments, we use the 306 possible pairings of the activity recognition datasets. However, this time we vary the number of days of labeled target data from 1 to 30. Figure 7 shows the results. Clearly, adding more labeled target data is initially beneficial. However, the increase in accuracy begins to level off after approximately ten days of labeled target data. The increase in recall appears to peek between five and ten days of labeled target data after which point the recall score declines slightly. This may indicate that having too much labeled data causes *IFSR* to overfit the data.

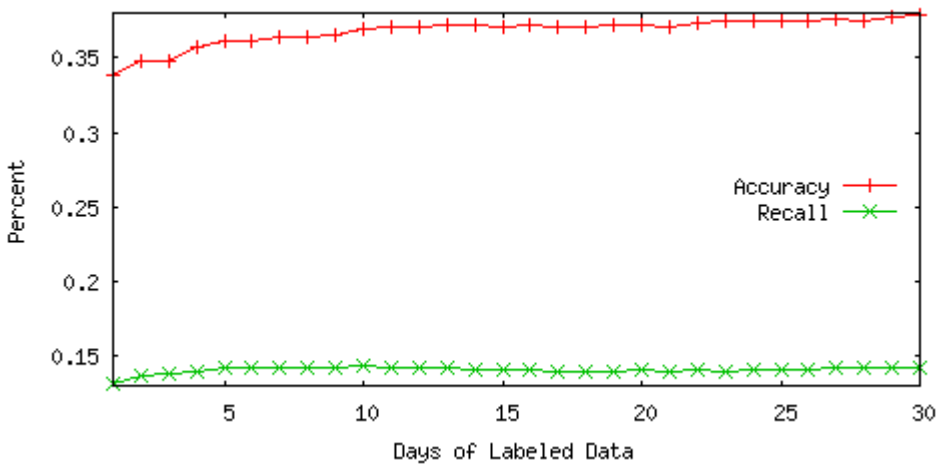


Figure 7: IFSR accuracy and recall scores as the amount of labeled target data used to make the mapping from target feature space to source feature space increases. Accuracy continues to show improvement with the increase of labeled target data while the recall score appears to peek with between five and ten days of labeled data in the target domain

5.6 *ELFSR Comparison*

Having examined the trade-off between the computational complexity and the performance results of the GAFSR, GrFSR, and IFSR techniques, the remaining results focus on combining multiple source datasets using ELFSR. In these experiment we have used ELFSR with the IFSR technique but GAFSR or GrFSR could be used as well. We consider different voting and stacking ensemble techniques which utilize data from multiple source datasets. IFSR-Maj refers to using ELFSR in a majority voting ensemble. IFSR-Sum refers to using ELFSR in a summed probability ensemble. IFSR-Bayes and IFSR-Tree refer to using ELFSR in a stacked ensemble using a naïve Bayes classifier or Decision Tree classifier, respectively, as the ensemble learning algorithm. We include several additional baseline techniques here. *Self* uses a naïve Bayes classifier which has been trained on the target dataset using 3-fold cross-validation and all 30 days of labeled data. *Combined* combines all of the source domain data into one big dataset with sensor mappings being manually defined by location. The naïve Bayes classifier is trained on all of the source data and then tested on the target data. The ensemble techniques each train one naïve Bayes classifier per source dataset and the ensemble is then tested on the target domain. Each stacking ensemble is trained using one day's worth of labeled data in the target domain.

Figure 8 shows the results using the voting ensemble techniques while Figure 9 shows the results using the stacking ensemble techniques. In neither case do we attempt to select the best source datasets: we simply use all available source datasets.

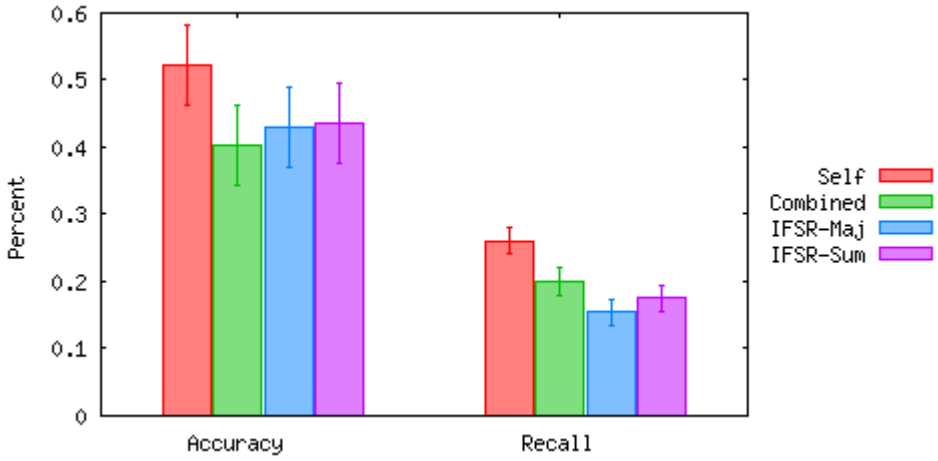


Figure 8: Classification accuracy and recall on the target domain using multiple source domains with a voting ensemble. *Self* and *Combined* provide baseline comparisons. Self is the result when the source and target dataset are the same and uses the all the labeled target data, while *Combined* uses the mappings provided by a domain expert to build a generic classifier. Matching the performance of *Combined* is a positive result.

The *IFSR* voting ensembles perform comparably to the combined dataset. This is consistent with the previous results where *IFSR* performs comparably to the *Manual* technique. The trade-off is where the human effort is required. The combined dataset requires a manually-mapped specification while the *IFSR* voting ensembles require a small amount of labeled data in the target domain.

The performance of the *IFSR* stacking ensembles stand out above the rest. Both stacking ensembles achieve higher performance in terms of the accuracy and recall scores than the combined dataset or the *Self* classifier. It does this using only a single day's worth of labeled data and no manual mapping is required. The *Self* approach uses nearly 30 days of labeled data and is trained and tested on the same dataset (with cross-validation), while the Combined approach uses no labeled data in the target domain but requires a manual mapping to be specified.

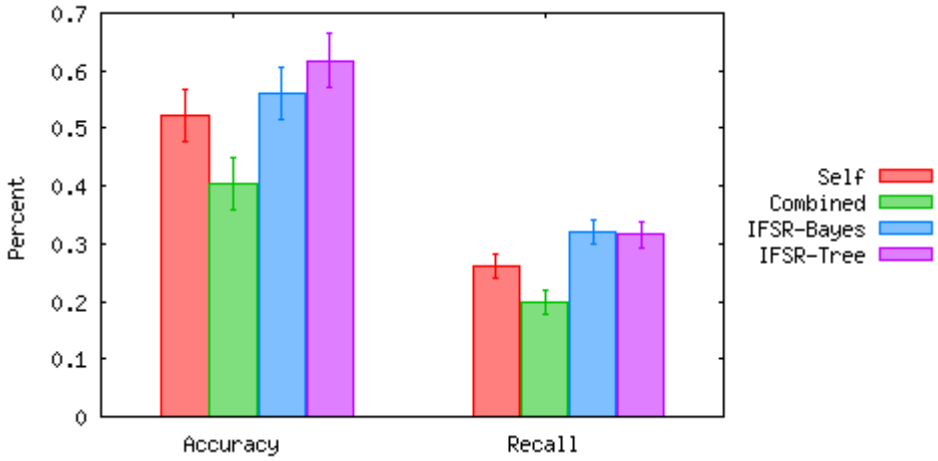


Figure 9: Classification accuracy and recall on the target domain using multiple source domains with stacking ensembles. *Self* and *Combined* provide baseline comparisons. Self is the result when the source and target dataset are the same and uses the all the labeled target data, while *Combined* uses the mappings provided by a domain expert to build a generic classifier. The performance of *IFSR-Bayes* and *IFSR-Tree* both manage to beat these baselines representing a considerable gain for the transfer learning techniques.

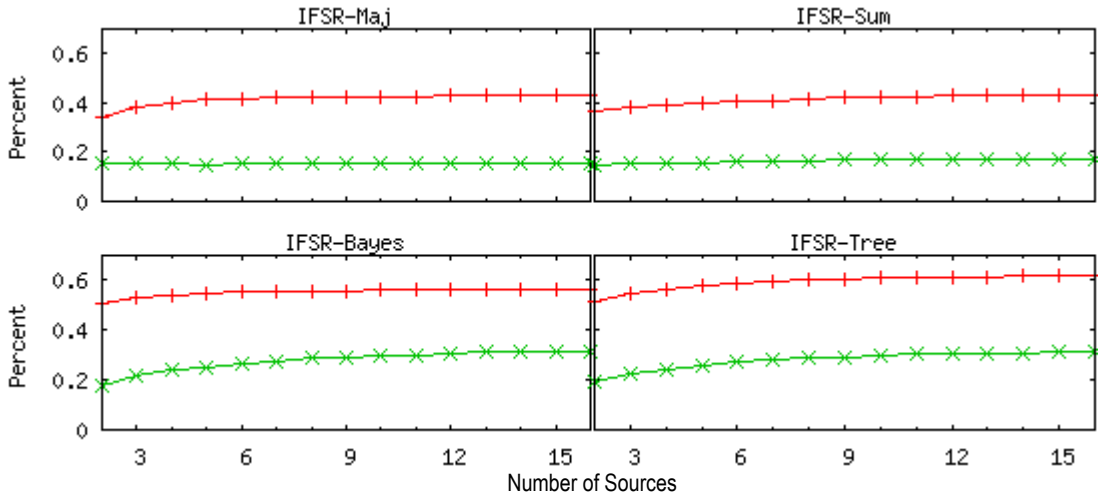


Figure 10: Learning curve for the ensemble classifiers where the number of source classifiers ranges from 2 to 16. Each ensemble technique quickly improves with more source classifiers but the performance improvements then begin to level off.

5.7 *ELFSR Learning Curve*

In addition to comparing the performance of *ELFSR* against other baseline techniques we also consider how the number of source datasets affects the performance achieved by the techniques. Figure 10 shows the learning curve for each ensemble technique as the number of source datasets increases. For *IFSR-Sum, IFSR-Bayes*, and *IFSR-Tree*, the performance increases with an increasing number of datasets. Most of the improvement is achieved within the first seven datasets, after which performance improvement tapers off. For *IFSR-Maj*, the accuracy performance improves with an increasing number of datasets, but the recall performance remains almost constant regardless of the number of datasets. This illustrates the fact that important distinguishing information is being discarded by the majority voting scheme.

6 Conclusions

In this paper we present novel heterogeneous transfer learning techniques for use in recognizing activities between different smart home environments. These techniques, transfer knowledge between domains with different feature spaces without using typical co-occurrence data. The datasets we tested on also had different marginal probability distributions on the domains, and different conditional probabilities. This makes the difference between source and target datasets greater than many previously attempted transfer learning problems. The key insight allowing these techniques to work is that by mapping the target features to the source features we are able to reuse an existing hypothesis to guide the search for 'good' maps. The proposed techniques can be generalized and applied to any heterogeneous transfer learning problem where labeled target data exists. The techniques are compatible with most other transfer learning techniques and could be applied as a pre-processing step to obtain a common feature space before applying traditional domain adaptation techniques.

Ensemble Learning via Feature-Space Remapping is introduced to combine multiple source datasets and achieve even greater classification accuracy. Using ELFSR we are able to outperform a classifier which has been trained and tested exclusively in the target domain using a full thirty days of labeled data, while ELFSR uses just a singled day of labeled data in the target domain.

There are still many open research questions to pursue, including avoiding negative transfer effects and identifying the best sources for transfer. An additional future direction involves the combining of multiple dimensions. The techniques we have explored generate a many-to-one mapping of target dimensions to source dimensions. We suggest exploring additional ways of combining multiple dimensions as well as exploring enforcing a one-to-one mapping or relaxing the mapping to allow a many-to-many mapping. Other challenges remain as well such as handling concurrent or interleaved activities. While there is still much research to be done, GAFSR, GrFSR and FSR are all promising new techniques to improve the transfer of knowledge between domains which will in turn lead to more robust activity recognition systems and learning systems in general.

References

Arnold, A.; Nallapati, R. & Cohen, W. (2007), "A Comparative Study of Methods for Transductive Transfer Learning", in *Seventh IEEE International Conference on Data Mining Workshops*, pp. 77 -82.

Barnett, S. & Ceci, S. (2002), "When and where do we apply what we learn?: A taxonomy for far transfer", *Psychological bulletin* Vol. 128 No. 4, pp. 612-637.

Blitzer, J.; Dredze, M. & Pereira, F. (2007), "Biographies, Bollywood, Boom-boxes and Blenders: Domain Adaptation for Sentiment Classification", in *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistic*.

Blitzer, J.; McDonald, R. T. & Pereira, F. (2006), "Domain Adaptation with Structural Correspondence Learning", in *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, pp. 120-128.

Breiman, L. (1996), "Bagging predictors", *Machine Learning* Vol. 24, pp. 123-140.

Byrnes, J. (1996), *Cognitive development and learning in instructional contexts*, Allyn and Bacon, Boston.

Cook, D. J.; Feuz, K. D. & Krishnan, N. C. (2012), "Transfer Learning for Activity Recognition", *Knowledge and Information Systems* Vol. 36, pp. 537--556.

Dai, W.; Chen, Y.; Xue, G.-R.; Yang, Q. & Yu, Y. (2008), "Translated learning: Transfer learning across different feature spaces", in *Advances in Neural Information Processing Systems*, pp. 353--360.

Dai, W.; Yang, Q.; Xue, G.-R. & Yu, Y. (2008), "Self-taught clustering", in *Proceedings of the 25th international conference on Machine learning*, ACM, New York, NY, USA, pp. 200--207.

Daumé, III, H. & Marcu, D. (2006), "Domain adaptation for statistical classifiers", *Journal of Artificial Intillegence Research* Vol. 26 No. 1, pp. 101--126.

Dietterich, T. G. (2000), "Ensemble Methods in Machine Learning", in *Proceedings of the First International Workshop on Multiple Classifier Systems*, Springer-Verlag, pp. 1--15.

Elkan, C. (2001), "The foundations of cost-sensitive learning", in *Proceedings of the 17th international joint conference on Artificial intelligence*, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, pp. 973--978.

Freund, Y. & Schapire, R. E. (1997), "A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting", *Journal of Computer and System Sciences* Vol. 55 No. 1, pp. 119 - 139.

Goldberg, D. E. & Holland, J. H. (1988), "Genetic algorithms and machine learning", *Machine learning* Vol. 3 No. 2, pp. 95--99.

Hansen, L. & Salamon, P. (1990), "Neural Network Ensembles", *IEEE Transactions on Pattern Analysis and Machine Intelligence* Vol. 12 No. 10, pp. 993-1001.

van Kasteren, T.; Englebienne, G. & Krцse, B. (2010), "Transferring Knowledge of Activity Recognition across Sensor Networks", in *Pervasive Computing*, Springer Berlin / Heidelberg, pp. 283-300.

van Kasteren, T.; Englebienne, G. & Krцse, B. (2008), "Recognizing activities in multiple contexts using transfer learning", in "AAAI AI in Eldercare Symposium".

Lafferty, J. & Zhai, C. (2001), "Document language models, query models, and risk minimization for information retrieval", in *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*, pp. 111--119.

Mitchell, M. (1998), *An Introduction to Genetic Algorithms (Complex Adaptive Systems)*, A Bradford Book.

Mitchell, T. M. (1997), "Bayesian Learning", *Machine Learning*, McGraw-Hill, Inc., New York, NY, USA, pp. 154 -- 200.

Pan, S. & Yang, Q. (2010), "A survey on transfer learning", *Knowledge and Data Engineering, IEEE Transactions on* Vol. 22 No. 10, pp. 1345--1359.

Pan, W.; Zhong, E. & Yang, Q. (2012), "Transfer learning for text mining" *Mining Text Data*, Springer, pp. 223--257.

Prettenhofer, P. & Stein, B. (2011), "Cross-lingual adaptation using structural correspondence learning", *ACM Transactions on Intelligent Systems and Technology (TIST)* Vol. 3 No. 1, pp. 13.

Rashidi, P. & Cook, D. (2011), "Activity knowledge transfer in smart environments", *Pervasive and Mobile Computing* Vol. 7 No. 3, pp. 331-343.

Rashidi, P. & Cook, D. (2010), "Multi home transfer learning for resident activity discovery and recognition", in *KDD Knowledge Discovery from Sensor Data*, pp. 56--63.

Russell, S. J. & Norvig, P. (2010), *Artificial Intelligence - A Modern Approach (3rd ed.)*, Pearson Education.

Thorndike, E. & Woodworth, R. (1901), "The influence of improvement in one mental function upon the efficiency of other functions", *Psychological review* Vol. 8 No. 3, pp. 247-261.

Thrun, S. (1996), *Explanation-based neural network learning: A lifelong learning approach*, Kluwer Academic Publishers.

Thrun, S. & Pratt, L. (1998), *Learning to learn*, Kluwer Academic Publishers.

Vilalta, R. & Drissi, Y. (2002), "A Perspective View and Survey of Meta-Learning", *Artificial Intelligence Review* Vol. 18, pp. 77-95.

Wang, Z.; Song, Y. & Zhang, C. (2008), "Transferred Dimensionality Reduction", in *Machine Learning and Knowledge Discovery in Databases*, Springer Berlin / Heidelberg, pp. 550-565.

Wolpert, D. H. (1992), "Stacked generalization", *Neural Networks* Vol. 5, pp. 241--259.

Xian-ming, L. & Shao-zi, L. (2009), "Transfer AdaBoost learning for action recognition", in *IEEE International Symposium on IT in Medicine Education*, pp. 659 -664.

Yao, Y. & Doretto, G. (2010), "Boosting for transfer learning with multiple sources", in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1855--1862.