Exploiting kernel-based feature weighting and instance clustering to transfer knowledge across domains

Jafar TAHMORESNEZHAD, Sattar HASHEMI*
Electrical and Computer Engineering School, Shiraz University, Shiraz, Iran

Received: 27.03.2015 • Accepted/Published Online: 17.12.2015 • Final Version: 24.01.2017

Abstract: Learning invariant features across domains is of vital importance to unsupervised domain adaptation, where classifiers trained on the training examples (source domain) need to adapt to a different set of test examples (target domain) in which no labeled examples are available. In this paper, we propose a novel approach to find the invariant features in the original space and transfer the knowledge across domains. We extract invariant features of input data by a kernel-based feature weighting approach, which exploits distribution difference and instance clustering to find desired features. The proposed method is called the kernel-based feature weighting (KFW) approach and benefits from the maximum mean discrepancy to measure the difference between domains. KFW uses condensed clusters in the reduced domains, the domains that do not contain variant features, to enhance the classification performance. Simultaneous use of feature weighting and instance clustering increases the adaptation and classification performance. Our approach automatically discovers the invariant features across domains and employs them to bridge between source and target domains. We demonstrate the effectiveness of our approach in the task of artificial and real world dataset examinations. Empirical results show that the proposed method outperforms other state-of-the-art methods on the standard transfer learning benchmark datasets.

Key words: Transfer learning, unsupervised domain adaptation, feature weighting, instance clustering, maximum mean discrepancy

1. Introduction
There are some examples [1–3] in the field of artificial intelligence that do not conform to the general assumption of standard machine learning. This leads to an issue known as the domain shift problem [4], where the training and test sets come from different distributions. For example, a developed Android application that has been trained with LabelMe [5] and ImageNet [6] datasets could not classify objects in captured images with a mobile phone camera [2]. In fact, the distribution and properties of the test and train sets show great differences. Moreover, generating labeled samples to learn a new model is very costly and time-consuming. Techniques addressing the challenge have been investigated including domain adaptation, covariate shift, and transfer learning.

This study sets to provide an efficient encounter with the problem of distribution difference across domains. There has been a plethora of recent publications addressing the same issue known as the two-sample or homogeneity problem. Borgwardt et al. proposed to test whether distributions $s$, the source domain distribution, and $t$, the target domain distribution, are different by finding a smooth function that is large

*Correspondence: s_hashemi@shirazu.ac.ir
on the points drawn from $s$ and small (as negative as possible) on the points from $t$ [7]. The mean function values of the projected domains (by the smooth function) are calculated and their difference is used as the test statistic. If the calculated value shows a large difference, the samples probably have different distributions. This statistic is called the maximum mean discrepancy (MMD).

However, in order to find the invariant features across domains, we propose a novel kernel-based feature weighting (KFW) approach with two main objectives: (1) to decrease the distribution distance of the source and target data in the reduced domains; and (2) to cluster the instances of the same classes in the resulting domains. The former objective is achieved by MMD, which is a nonparametric approach to compare distributions. The latter objective is accomplished by searching for the reduced domains that force instances with the same labels to form more condensed clusters. This can be obtained by minimizing the distance between the samples of each class and their means.

**Contributions:** In this work we contribute to the solving of the domain shift problem and show i) how to formulate the problem of transfer learning as a feature weighting problem; ii) how to construct compact clusters to formulate the optimization problem; iii) how to automatically identify different types of features across the source and target domains; iv) how to solve the optimization problem; v) and finally, how our method outperforms other feature-based state-of-the-art transfer learning approaches on the artificial and real world benchmark datasets.

**Organization of the paper:** In the next section, we relate our approach to the existing research on transfer learning. Section 3 describes the theoretical background behind the proposed approach. In section 4 we introduce our proposed method and present the main algorithm. Our method is evaluated on a dataset that has been designed specifically to study the problem of domain shift. Then we show the results of classifier adaptation on several challenging shifts in sections 5 and 6. These are followed by the conclusion and future work in the last section.

### 2. Related Work

Transfer learning and domain adaptation [3, 4] are challenging research areas in recent years and they have been comprehensively studied from various perspectives, including natural language processing [10, 14], statistics and machine learning [12, 15], and recently computer vision [16–19]. Pan et al. [4] presented a complete survey of cross-domain learning methods, and discussed the different applications of transfer learning.

The available domain adaptation approaches are divided into three main categories: (1) instance-based approaches, (2) model-based approaches, and (3) feature-based approaches. Instance-based approaches [10, 20] emphasize sample selection or reweighting of the source data according to their difference from the target data. Kernel-based feature mapping with ensemble (KMapEnsemble) [21] is an adaptive kernel- and sample-based method that maps the marginal distribution of the source and target data into a shared space, and exploits a sample selection method to reduce conditional distribution across domains. Our proposed method has essential differences from KMapEnsemble. KFW is a feature-based transfer learning approach that does not filter source domain samples in the reduced domain. Moreover, KFW finds invariant features in the original feature space, and does not map input data into a shared space and preserves the original properties of data. Thus, KFW will not have entropy increase drawback, like KMapEnsemble, due to data mapping input data into a common representation.

Model-based domain adaptation approaches [22, 23] discover an adaptive classifier that performs well on the target data. In these models, the learned classifier transfers model parameters from the source to the target
domain without any change in the feature space. Most of the available methods in this category find an adaptive classifier using a support vector machine (SVM), and use it for semisupervised domain adaptation problems [16, 24]. The model-based approaches are dependent on a specific model and they are affected by the model properties, whereas KFW is a model-independent method that preprocesses the shifted data and generalizes it to most of the models.

The third class of transfer learning approaches is feature-based methods [8–11]. Transfer component analysis (TCA) [12] is a dimensionality reduction and feature weighting approach that employs MMD to reduce differences between source and target domains. TCA finds transfer components across domains to adapt source and target domains in a reproducing kernel Hilbert space (RKHS). KFW finds a shared invariant space in original space and minimizes both marginal and conditional distribution differences in reduced domain. However, TCA only concentrates on marginal distribution between source and target domains and explicitly does not reduce the conditional distribution. Moreover, KFW exploits source domain labels to cluster the same class instances in the reduced domains, whereas TCA is an unsupervised method and does not employ the label information of input data.

Feature selection by maximum mean discrepancy (f-MMD) [13] is another feature-based dimensionality reduction approach that exploits MMD to measure differences between domains. Despite the good performance of f-MMD on different datasets, it performs domain adaptation in a fully unsupervised manner. However, KFW measures the difference between source and target domains in a supervised solution and tags the available features in the input space into variant and invariant. The features that contribute to the variation across domains are referred to as the variant features and those that contribute to the closeness of distributions of the source and target domains as the invariant features.

Our work belongs to the feature-based category [12, 13, 25, 26], which finds a shared feature space across domains [17, 27–29]. The achieved space has lower discrepancy and also preserves the important properties of input data. Moreover, it reduces the marginal distribution between domains due to exploiting kernel-based feature selection. In this paper, we propose a joint feature weighting and instance clustering method that employs domain invariant clustering to discriminate between various classes.

3. Maximum mean discrepancy

In this work, we intend to measure dissimilarity between probability distribution of the source domain, \( s \), and the target domain, \( t \). For this purpose, we exploit MMD, which is a nonparametric measure to compare the distribution difference of source and target domains by mapping the data into a rich reproducing kernel Hilbert space. Given two distributions \( s \) and \( t \) and following [7], MMD is formulated as:

\[
MMD(X_s, X_t, F) = \sup(E[f(x_s)] - E[f(x_t)]),
\]

where \( X_s \) and \( X_t \) are the source and target datasets, respectively, and \( E[f(x_s)] \) and \( E[f(x_t)] \) are expectation under distributions of \( s \) and \( t \), in turn. \( F \) is defined as a rich class of functions, e.g., unit ball in the universal RKHS. However, \( MMD(X_s, X_t, F) \) tends towards zero if and only if \( s = t \). The main idea is that if the feature means of domains under RKHS are close to each other, the distribution of domains will be close in the original space [30]. \( X_s = \{x^1_s, x^2_s, \ldots, x^n_s\} \) and \( X_t = \{x^1_t, x^2_t, \ldots, x^n_t\} \) are defined as the observations drawn
i.i.d. from \( s \) and \( t \), respectively. Following [12] an empirical estimate of MMD can be calculated as

\[
D(X_s,X_t) = \left\| \frac{1}{n} \sum_{i=1}^{n} \Phi(x^i_s) - \frac{1}{m} \sum_{j=1}^{m} \Phi(x^j_t) \right\|_H
\]

(2)

where \( n \) and \( m \) are the number of source and target samples, respectively, and \( \Phi(x) \) is the feature map defined as \( \Phi(x): X \rightarrow H \), where \( H \) is a universal RKHS. If the universal kernel associated with this mapping is defined as \( k(z_i, z_j^T) = \Phi(z_i)\Phi(z_j^T) \), according to \( (u-v) = ((u-v)^2)^{\frac{1}{2}} = (u^2 + v^2 - 2uv)^{\frac{1}{2}} \), Eq. 2 can be rewritten as

\[
D(X_s,X_t) = \left( \sum_{i=1}^{n} \sum_{j=1}^{n} \frac{k(x^i_s, x^j_s)}{n^2} + \sum_{i=1}^{m} \sum_{j=1}^{m} \frac{k(x^i_t, x^j_t)}{m^2} - 2 \sum_{i=1}^{n} \sum_{j=1}^{m} \frac{k(x^i_s, x^j_t)}{nm} \right)^{\frac{1}{2}}
\]

(3)

In a nutshell, MMD between the distributions of two sets of observations is equivalent to the distance between the sample means in a high-dimensional feature space [31].

4. Proposed approach

In this paper, we aim to find a solution for the problem of unsupervised domain adaptation using kernel-based feature weighting and instance clustering. Figure 1 shows the flowchart of KFW, where both domains are composed of variant and invariant features. The goal of KFW is to discriminate the features based on their effectiveness in the variation between domains. The optimization problem calculates the optimal weight matrix, \( W^* \), based on the difference of domains and clustering domain instances.

![Figure 1. The schematic flowchart of KFW. \( W^* \) is the optimal weight vector that is obtained from solving the optimization problem. \( \lambda \) is a threshold parameter, which distinguishes the invariant features for target classifier.](image)

4.1. Kernel-based feature weighting (KFW)

The invariant features are common across domains and also have most statistics and properties of the input data. Thus, we find invariant features that minimize the distance between the source and target domains. More specifically, we define \( W \) as a diagonal weight matrix that corresponds to the weights of the features. Let us define \( X_s \) as an \( n \times d \) matrix containing \( n \) samples and \( d \) features from the source domain and \( X_t \) as an \( m \times d \)
matrix containing $m$ samples and $d$ features from the target domain. Our aim is to find the matrix $W$ in such a way that the reduced domains have similar distributions.

$$W^* = \arg\min_W D^2(X_s, X_t)$$

s.t. $\|\text{diag}(W)\|_2 = 1$ and $W > 0,$

where $\text{diag}(W)$ is the diagonal of the weight matrix and $D(.,.)$ denotes the difference between domains. The constraints control the range of $W$ where the first constraint restricts the size of weights and the second one lets $W$ have only positive values. Following [13], Eq. 4 can be rewritten in the form below using the kernel trick, i.e. $k(z_i, z_j^T) = \Phi(z_i)\Phi(z_j^T),$ where $k$ is a positive definite kernel and $\text{tr}(.)$ denotes the trace of the matrix:

$$W^* = \arg\min_w \text{tr}(KL)$$

s.t. $\|\text{diag}(W)\|_2 = 1$ and $W > 0,$

where $K = \begin{bmatrix} K_{s,s} & K_{s,t} \\ K_{t,s} & K_{t,t} \end{bmatrix} \in R^{(n+m)\times(n+m)},$ and $L = \begin{bmatrix} L_{s,s} & L_{s,t} \\ L_{t,s} & L_{t,t} \end{bmatrix} \in R^{(n+m)\times(n+m)}$ are composite kernel and coefficient matrices, respectively. Moreover, $K_{s,s}, K_{t,t},$ and $K_{s,t}$ are kernel matrices that have been defined by $k$ on the source, target, and cross domains respectively. In addition, the value of coefficients is calculated by $L_{s,s} = \frac{1}{n^2},$ $L_{t,t} = \frac{1}{m^2},$ and $L_{s,t} = \frac{1}{nm}$ [31]. Each element in $K$ is computed using the kernel function; thus, they depend on $W; e.g.,$ the polynomial kernel $K_{s,s}$ with the degree $p$ is calculated by $K_{s,s} = (1 + xWx)^p.$

Considering the simplified Gaussian kernel function as $G(u, v) = e^{-\frac{(u-v)^T(u-v)}{s}},$ Eq. 3 can be rewritten as MMD expression in terms of the Gaussian kernel function:

$$D^2(X_s, X_t) = \frac{1}{n^2} \sum_{i=1}^{n} \sum_{j=1}^{n} e^{-\frac{(x_i^T-x_j^T)(x_i^T-x_j^T)}{s}} + \frac{1}{m^2} \sum_{i=1}^{m} \sum_{j=1}^{m} e^{-\frac{(x_i^T-x_j^T)(x_i^T-x_j^T)}{s}}$$

$$- \frac{2}{nm} \sum_{i=1}^{n} \sum_{j=1}^{m} e^{-\frac{(x_i^T-x_j^T)(x_i^T-x_j^T)}{s}}$$

The Gaussian kernel is a universal kernel; however, in practice, nonuniversal kernels show more appropriate results in measuring MMD [7]. The polynomial kernel of degree two is a more general kernel that has little idiosyncratic effect on the experiments, unlike the Gaussian kernel. Therefore, replacing the Gaussian kernel with a polynomial kernel in the objective function yields

$$D^2(X_s, X_t) = \frac{1}{n^2} \sum_{i=1}^{n} \sum_{j=1}^{n} (1 + x_i^T x_j^T)^2 + \frac{1}{m^2} \sum_{i=1}^{m} \sum_{j=1}^{m} (1 + x_i^T x_j^T)^2$$

$$- \frac{2}{nm} \sum_{i=1}^{n} \sum_{j=1}^{m} (1 + x_i^T x_j^T)^2$$

(7)

4.2. Instance clustering in reduced domains

The weight matrix $W$ determines the category of each feature based on the weight that it takes from the optimization problem. So far, in finding $W$ only the difference of distributions of domains is considered,
Algorithm 1 The optimization problem of KFW

1: \texttt{cvx_begin}
2: \texttt{variable W(d,d) diagonal}
3: \texttt{K = CalculatePolynomialKernelFunction(Xs, Xt, W, p)};
4: \texttt{L = CalculateCoefficientMatrix(n, m)};
5: \texttt{Q = CalculateMatrixQ(class labels, Xs, mean values)};
6: \texttt{W = minimize(trace(KL) + \beta \times trace(W^TQ^TW))};
7: \texttt{subject to}
8: \texttt{W > 0}
9: \texttt{\|diag(W)\| = 1}
10: \texttt{cvx_end}

whereas the distribution difference alone is not enough to classify the reduced domain and determine the target labels. Indeed, we are to transfer knowledge from the source to the target domain. In order to achieve this goal and in addition to reduce the discrepancy between domains, we need to minimize the within-class scatter to form more compact instance clusters in the reduced domains where the source and target data have similar distributions. The within-class scatter, \(S_W\), aggregates the same class instances around its mean, and assigns higher weights to the features that contribute to the classification performance. \(S_W\) is defined as follows:

\[
S_W = tr((WQ)^T)
\]  \quad (8)

where \(Q \in \mathbb{R}^{n \times d}\) is a zero-mean matrix that contains the distance of each instance from its class mean in the source domain, and for each \(i = 1, \ldots, n\) and \(c = 1, \ldots, C\) the value of \(Q_i\), the \(i^{th}\) row of matrix \(Q\), is calculated by \(Q_i = x_{sc}^i - \mu_c\). \(C\) is the number of domain classes and \(\mu_c\) denotes the mean of samples in class \(c\). Therefore, the within-class scatter can be simplified to \(S_W = tr((WQ)^T(WQ)) = tr(Q^TWWQ)\). Moreover, since \(tr(A) = tr(A^T)\), we can rewrite \(S_W = tr(W^TQ^TW)\).

In this way, in finding \(W\), the weights are adjusted in a manner that the instances from the same class have lower distances from the class mean. Thus, each instance falls into a more compact cluster; hence, the classification performance increases. Our reformulated optimization problem is

\[
W^* = \argmin_w tr(KL) + \beta S_W
\]  \quad (9)

s.t. \(\|\text{diag}(W)\| = 1\) and \(W > 0\),

where \(\beta\) is the regularizer whose value is determined 0.25 based on the numerous experiments on real and artificial experiments.

Since Eq. 9 is a quadratically constrained quadratic program (QCQP), it should be solved using a QCQP solver such as CVX (abbreviation for ConVeX). CVX is a strong tool for convex function optimization [32]. Algorithm 1 shows KFW, where \(W^*\) contains the optimized weights. Because the weight values in matrix \(W^*\) are very small, \(W^*\) is normalized before feature discrimination. The weight of each feature classifies it as either variant or invariant. \(\lambda\) is the threshold value, which is determined in an unsupervised manner by experiments. In the next section, we will show how the value of \(\lambda\) is determined. The features with weights more than \(\lambda\) are considered as invariant, and clearly the features with weights less than \(\lambda\) are defined as variant.

4.3. Computational and space complexity

We analyze the computational complexity of Algorithm 1 using the big \(O\) notation. We denote \(d\) the number of features, and \(n\) and \(m\) the number of source and target instances, respectively. The computational cost
is detailed as follows: \( O(n^2d^2 + nm^2 + m^2d^2) \) for composing kernel matrix \( K \), i.e. Line 3; \( O((n + m)^2) \) for constructing coefficient matrix \( L \), i.e. Line 4; \( O((n + m)^3 + 3nd^2 + (n + m)^{4.5} + d) \) for all other steps including optimization problem and feature categorization, i.e. Line 5. In summary, the overall computational complexity of Algorithm 1 is \( O((n + m)^4) \).

We also evaluate the space complexity of Algorithm 1 using the big \( O \) notation. The space cost is detailed as follows: \( O(d) \) for defining diagonal weight matrix \( W \), i.e. Line 2; \( O((n + m)^2) \) for composing kernel matrix \( K \), i.e. Line 3; \( O((n + m)^2) \) for constructing coefficient matrix \( L \), i.e. Line 4; \( O(nd) \) for creating zero-mean matrix \( Q \), i.e. Line 5; \( O((n + m)^2 + d^2) \) for all other steps, including optimization problem solving, i.e. Line 6. In summary, the overall space complexity of Algorithm 1 is \( O((n + m)^2 + nd + d^2) \).

5. Experimental setup

5.1. Data description

Experiments are conducted on two real and two artificial, shifted, datasets where the Table shows the artificial data, which includes basic statistics, such as distribution, number of examples, and features. The number of instances is supposed to be the same across domains. A short description of the real and artificial datasets is given in the following sections.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>N</th>
<th>V</th>
<th># instances</th>
<th>Dist. of source</th>
<th>Dist. of target</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gau</td>
<td>10</td>
<td>40</td>
<td>300</td>
<td>Gaussian</td>
<td>Gaussian-Gaussian</td>
</tr>
<tr>
<td>UniPoi</td>
<td>40</td>
<td>10</td>
<td>600</td>
<td>Uniform</td>
<td>Uniform-Poisson</td>
</tr>
</tbody>
</table>

5.1.1. Artificial datasets

In order to generate the artificial datasets, the invariant and variant features should be sampled from different distributions to simulate the domain shift problem. The number of invariant and variant features is indicated by \( N \) and \( V \), respectively. The Table illustrates the artificial datasets in more detail. Dataset \( \text{Gau} \) is a shifted dataset composed of the source and target domains where the total number of features is 50. For both domains, \( N \) invariant features are sampled from \( N \) randomly picked distributions with zero mean and unit variance. For the source domain, \( V \) variant features are sampled from \( V \) randomly picked distributions with zero mean and unit variance. For the target domain, \( V \) variant features are sampled from \( V \) randomly picked distributions with shifted mean and unit variance.

Dataset \( \text{UniPoi} \) is generated similar to \( \text{Gau} \) with the difference that for both domains \( N \) invariant features and for the source domain \( V \) variant features are sampled from randomly picked \textit{uniform} distribution, and for the target domain \( V \) variant features are sampled from a randomly picked \textit{Poisson} distribution. In order to generate the class labels, we use the sign function that is applied to the weighted instances. The class labels are generated using \( r \) number of features randomly selected from the total number \( (d) \) of features. \( g \) is a \( d \)-dimensional weight vector drawn from a uniform distribution. Every element in \( g \) is set to zero only if it is not included in \( r \). Finally, the class labels \( (l) \) for data are generated by \( l = \text{sign}(g * x) \), where \( x \) is the input data.
5.1.2. Real datasets

WiFi localization and lung datasets are two real world datasets based on which the performance of KFW is evaluated. The first real evaluation is conducted on the task of indoor WiFi localization utilizing a sample dataset released in the 2007 IEEE ICDM Contest for transfer learning. The goal is to locate a WiFi gadget based on its received signal strength (RSS) qualities from different access points. The dataset contains some labeled WiFi information gathered in the time period A (the source domain) and a lot of unlabeled WiFi information gathered in the time period B (the target domain). In our experiments, we adjust the number of source domain instances to 621 and vary the number of target data between 50 and 250. Moreover, the number of features and locations of the environment are 99 and 247, respectively.

The second real evaluation is conducted on the lung dataset. The lung dataset contains 30 chest radiographs obtained from the JSRT database [33]. The number of selected features for each pixel is 10 using N-jets feature representation [34]. From available classes in the dataset, lung, rib, and background are selected. Each radiograph is digitized and normalized to a $32 \times 32$ matrix. In our experiments, in each iteration, one image is selected as the source domain and the rest as the target. The number of features in each domain is selected 10, the same as the number of features that explain each pixel.

5.2. Method evaluation

In our experiments, we compare the performance of KFW with that of the other two feature-based transfer learning approaches, TCA and f-MMD. All methods are evaluated based on their reported best results. The parameters of TCA and f-MMD are adjusted to 1 and 0.1, respectively, and they are fixed during the tests. Since KFW, TCA, and f-MMD are dimensionality reduction approaches, linear SVM and logistic regression classifiers are trained on the labeled source data for classifying the unlabeled target data. We choose these classifiers because they are linear and capture the contribution of each feature independently. In the next section, we show the performance of KFW against other feature-based transfer learning approaches.

6. Empirical results and discussion

In this section, we report the results of our experiments on the artificial and real datasets with respect to our contributions in the introduction section.

Our first experiment, considering the domain difference of the source and target data, shows how KFW distinguishes the variant and invariant features across domains. Figure 2 shows the weights assigned by KFW to each feature on the artificial datasets. The horizontal axis denotes the feature set and the vertical axis shows the weight assigned to each feature by KFW. As is clear from the plots, the variant and invariant features can be distinguished by a distinct margin. Although the width of the margin could be different based on the type of distribution across domains, there is a deterministic line that separates the feature space into different variant and invariant subspaces. In the next section, we will show the impact of parameter settings on the results of KFW.

6.1. Artificial datasets

As the next experiment, we design a test to compare the performance of our proposed approach with and without variant features. Indeed, we exploit linear SVM and logistic regression classifiers on the reduced and original domains to show the superiority of KFW in distinguishing and removing the variant features. Figure 3 illustrates the classification accuracy of linear SVM on the dataset reduced by KFW (invariant only) and the original dataset. The number of samples increases from 100 through 350 in order to examine the performance...
Figure 2. KFW assigns a weight to each feature, which discriminates variant and invariant features.

of the proposed approach with different number of instances. The number of samples increases the accuracy of the model in most cases. In fact, the kernel matrix $K$ will contain more samples and has more accurate estimation from variant and invariant features. According to Eq. 5, the size of matrix $K$ is directly dependent on the number of source and target samples, and increases the execution time of the algorithm with respect to the discussions in previous sections.

Figure 3. The performance of KFW is evaluated on artificial datasets using linear SVM classifier.

In Figure 4, a logistic regression classifier has been used to model data. The results illustrate that the classification accuracy improves surprisingly with removing the variant features from the dataset. However, the amount of improvement is different according to the distribution of domains and the number of features. In other words, KFW shows better performance in some datasets due to the increase in the number of invariant features compared to the variant features, e.g. UniPoi dataset. In this case, the probability of information loss due to feature selection is low. The number of samples increases model accuracy in most cases. In fact,
according to Eq. 9, increasing the number of samples lets the clusters contain a larger number of instances and the estimation of mean value is done with more precision.

Figure 4. The performance of KFW is evaluated on artificial datasets using logistic regression classifier.

From another view, the performance of KFW is significantly dependent on the number of samples. We reduce the distance of domains according to the kernel function on the source and target data. In this way, exploiting more knowledge helps find the variant and invariant features accurately. Thus, it is not unexpected to have better performance in domains with a larger number of samples. Figures 3 and 4 clearly show that the accuracy of KFW has an increasing trend corresponding to the increase in the number of samples. This growth in logistic regression classifier is tangible; however, there is no significant difference between the two classifiers.

Figure 5. Accuracy of feature-based transfer learning methods on artificial datasets.

For the third artificial experiment, we design a dataset with different number of instances and 30 features, where the numbers of variant and invariant features are equal. In this experiment, half the features are selected randomly to generate the class labels using the procedure mentioned previously. The distribution of variant
and invariant features is Gaussian with different means and the same variance. Actually, we impose the shift synthetically on the source and target data.

In order to compare the performance of KFW with other state-of-the-art transfer learning approaches, we plot Figures 5a and 5b, where the number of samples varies from 100 through 350. In fact, all three methods reduce the dimension of domains for classification. Nevertheless, KFW finds a robust and effective reduced domain that decreases distance across domains. Moreover, KFW exploits the condensed clusters in the resulting domain to select the features that have more dependence on the class label. In general, KFW outperforms TCA and f-MMD in all cases; however, with increasing number of samples, the accuracy of f-MMD and KFW increases while TCA shows fluctuation in results. Despite the fact that TCA and f-MMD are feature-based approaches, they perform domain adaptation fully unsupervised and do not exploit the source domain labels; thus, they miss an important part of information in the learning task. Therefore, the performance of TCA and f-MMD degrades unexpectedly.

In general, KFW almost has a growing trend in performance, but in some cases its accuracy degrades with respect to the number of samples. For example, in Figure 5b, when the number of instances increases from 100 to 150, accuracy decreases to 84.8 due to feature elimination. In fact, some class labels have a strong relation with the variant features and their elimination decreases the performance of the model.

6.2. Real datasets

In this section, we show that how KFW reduces the distance between source and target domains in real world benchmark datasets. Figure 6 compares the performance of KFW with other feature-based transfer learning methods and also without exploiting transfer learning. In an indoor WiFi localization dataset (Figure 6a), the task is to identify the labels of WiFi data collected during time period B according to the data collected in time period A. Each experiment is repeated 10 times and the average error distance (AED) is calculated according to the following relation:

\[
AED = \frac{\sum_{(x_i, y_i) \in D} |f(x_i) - y_i|}{N}
\]
where $x_i$, $f(x_i)$, and $y_i$ are the vector of RSS values, predicted location, and corresponding true location, respectively. Following [12] we used a source dataset of 621 samples, and vary the number of target data from 50 through 250. The number of instances has a negative influence on the error rate of KFW. In fact, the generalization of KFW with more instances and without the variant features is outstanding. However, TCA has the worst performance, because it projects the data into the latent space without considering the relation between the features and class labels. Indeed, TCA finds the transfer components based on the variance of data across domains. Nevertheless, f-MMD has better performance compared to TCA, where it transfers the knowledge in the original space and does not project the domains into a latent space.

In general, KFW removes the variant features in the original feature space, and only exploits the invariant features to locate the true location of gadgets. In this way, KFW determines the weight of each feature according to its participation in the distance of domains, and only preserves the features with high weights.

Lung is the next real world dataset that is examined using KFW, TCA, and f-MMD. In this experiment, we aim to show that KFW could decrease the distance of domains better than other transfer learning approaches. Following [35], we used a source dataset of one image, and varied the number of target data from 4 through 29 images. The performance measurement is performed based on the accuracy of different methods.

Figure 6b shows the results on the lung dataset where the number of classes is three. As is clear from the figure, KFW outperforms TCA and f-MMD in all cases, and shows better performance. KFW decreases the distance of domains by exploiting MMD and finding domain invariant clusters. In this way, it finds a reduced space that has the main properties of input data and also classifies target samples accurately.

In addition, Figure 6b illustrates that f-MMD and TCA have close performances to each other, and in some cases TCA surpasses f-MMD in accuracy. In fact, when the number of images increases to 30, TCA shows better average on the results. However, both TCA and f-MMD find the shared domains in a fully unsupervised manner and do not exploit the source domain labels to obtain the invariant features. In general, KFW outperforms TCA and f-MMD in all cases and in some cases by a large margin.

The number of variant and invariant features in the real datasets is unspecified, and it is related to the nature of datasets. Indeed, we do not determine primarily the number of variant and invariant features in real datasets; however, KFW finds 34 variant and 65 invariant features in the WiFi dataset and 3 variant and 7 invariant features in the lung dataset. In this way, the size of reduced dimensions of source domain for the WiFi and lung datasets is $(621 \times 65)$ and $(1 \times 32 \times 32) \times 7$, respectively, where 621 and $(1 \times 32 \times 32)$ are the number of instances in the WiFi and lung datasets in turn.

### 6.3. Impact of parameter settings

KFW is evaluated with respect to different values of parameters to analyze its performance in different conditions. In general, we should tune the threshold parameter, $\lambda$, for KFW on different datasets. We report the results of KFW on $Gau$, $UniPoi$, $WiFi$, and $lung$ datasets.

Figure 7a illustrates the experiments on the $Gau$ dataset for evaluating parameter $\lambda$. We run KFW with varying values of $\lambda$. We report the classification accuracy of KFW with $\lambda \in [0.01, 0.9]$ on the $Gau$ dataset. The value of $\lambda$ determines the margin between the variant and invariant features. The plot indicates that in most cases increasing the value of $\lambda$ decreases the performance of KFW while the accuracy has a negative slope. Indeed, KFW shows better performance with low values of $\lambda$. In this way, $\lambda \in [0.05, 0.2]$ for the $Gau$ dataset is chosen.

Figure 7b shows the results for parameter $\lambda$ on the $UniPoi$ dataset. The figure denotes the classification
accuracy of KFW evaluated with $\lambda \in [0.01, 0.9]$ on the UniPoi dataset. As is clear from the plot, in most cases KFW shows acceptable results with small values of $\lambda$. Indeed, we choose $\lambda \in [0.05, 0.3]$ for the UniPoi dataset. In general, larger values of threshold parameter can give more importance to variant features and domain adaptation is not performed.

Figures 7c and 7d show the results for parameter $\lambda$ on the WiFi and lung datasets. The shown subfigures denote the average error/classification accuracy of KFW with $\lambda \in [0.01, 0.9]$ on the WiFi and lung datasets. As is clear from the plots, in most cases KFW shows acceptable results with small values of $\lambda$. Indeed, we choose $\lambda \in [0.05, 0.1]$ for the WiFi and $\lambda \in [0.05, 0.2]$ for the lung datasets. It is worth noting that in some datasets large values of $\lambda$ show good performance; however, the overall generalization of KFW with respect to the chosen interval is noteworthy. In this way, we choose a common value 0.1 for all datasets.

6.4. Convergence and time complexity

In this section, we also evaluate the convergence and time complexity of KFW compared to the state-of-the-art domain adaptation methods. KFW is a noniterative dimensionality reduction approach that finds invariant
features across domains and it converges in only one iteration. In this way, its main time complexity forms in solving the optimization problem; thus, the overall computational complexity of KFW is $O((n+m)^{4.5})$. TCA is a high-performing algorithm that converges in one iteration too, and its time complexity is $O(e(n+m)^2)$, where $e$ denotes the number of nonzero eigenvectors. The computational complexity of f-MMD is similar to that of KFW, i.e. $O((n+m)^{4.5})$, and it also converges in only one iteration. In general, all three methods converge in only one iteration, but TCA has better time complexity. However, since we construct the model in an off-line manner, time complexity does not possess much significance, and the main measure to compare the performance of different algorithms is their classification accuracy.

7. Conclusion and future work

Learning domain invariant features are critical for the problem of domain shift where the source and target instances follow different distributions. In this paper, we proposed a novel dimensionality reduction approach in the original feature space that distinguishes variant and invariant features across domains. Our aim is to assign the weight to each feature based on its contribution to the distance between domains where maximum mean discrepancy is exploited to measure distance across domains. Furthermore, the proposed method benefits from instance clustering to enhance the classification performance in the reduced domains. On benchmark tasks both artificial and real world, our method consistently outperforms other feature-based transfer learning methods. For future work, we plan to advance in this direction further, i.e. proposing KFW for multidomain settings.

References


