Transfer Learning with One-Class Data

Jixu Chen\textsuperscript{a,*}, Xiaoming Liu\textsuperscript{b}

\textsuperscript{a}GE Global Research, Niskayuna, NY 12309
\textsuperscript{b}Michigan State University, East Lansing, MI 48824

Abstract
When training and testing data are drawn from different distributions, most statistical models need to be retrained using the newly collected data. Transfer learning is a family of algorithms that improves the classifier learning in a target domain of interest by transferring the knowledge from one or multiple source domains, where the data falls in a different distribution. In this paper, we consider a new scenario of transfer learning for two-class classification, where only data samples from one of the two classes (e.g., the negative class) are available in the target domain. We introduce a regression-based one-class transfer learning algorithm to tackle this new problem. In contrast to the traditional discriminative feature selection, which seeks the best classification performance in the training data, we propose a new framework to learn the most transferable discriminative features suitable for our transfer learning. The experiment demonstrates improved performance in the applications of facial expression recognition and facial landmark detection.

Keywords: Transfer Learning, expression recognition, landmark detection

1. Introduction
A common assumption in traditional machine learning algorithms is that the training and testing data share the same distribution. However, this assumption may not hold in many real-world applications. When the distribution changes, most statistical models need to be retrained using the newly collected data. In order to reduce the burden of recollecting and relabeling training data, the transfer learning framework is introduced [1, 2, 3].

\*Corresponding author. Tel: (518) 387-5567 Fax: (518) 387-4136
\ Email address: chenji@ge.com (Jixu Chen)

Preprint submitted to Elsevier August 1, 2013
Figure 1: Illustration of transfer learning with one-class data in pain expression recognition. Traditional classifier (solid line) is learned from the training data of different subjects and applied on a new subject. Our algorithm takes a few one-class data samples (e.g., negative samples) of the new subject and learns a new classifier. Here, + and − denote positive data samples (e.g., pain expression) and negative data samples (e.g., non-pain expression) respectively. Different colors represent different subjects. ⊙ represents the negative data in a target domain (e.g., a new subject).

Transfer learning (TL) represents a family of algorithms that transfer the informative knowledge from a source domain\(^1\), where the training data is adequate, to a target domain, where the data is limited and follows a different distribution. For example, the concept of transfer learning has been explored extensively in speech recognition [4, 5]. While the speech recognizer is trained on a large training set, its performance on a new target speaker can be poor due to the variability of human voices. On the other hand, the speeches from different speakers share many similarities. A typical TL application is speaker-adaptation, which adapts the generic speech recognition model to a new target speaker using a small amount of data collected from that speaker. Similarly, in facial expression recognition it is beneficial to adapt a generically trained expression model to a new person through TL.

In this paper, we focus on the two-class classification problem. Conventional TL algorithms assume that data samples from both positive and negative classes

\(^1\)Following the definition in [1], a domain \(D\) consists of two components: a feature space \(\mathcal{X}\) and a marginal probability distribution \(P(x)\), where \(x \in \mathcal{X}\). Here, we focus on the transfer learning when the source and target domains have different distributions.
are available in the target domain [6]. In contrast, we study a new TL setting, where only one-class data (e.g., negative data) is available in the target domain. This setting is in sharp contrast from previous TL algorithms, but is not uncommon in real-world applications. For example, in pain expression recognition, as shown in Figure 1, a new subject has to enact the pain expression for the collection of the positive data in the target domain. This process is unnatural and cumbersome for the user, and this posed expression may be different from the spontaneous expression in the actual system execution. On the other hand, collecting the negative data (e.g., non-pain expression) of a new subject is much easier. Note that non-pain expression represents any natural expressions other than pain. The most common non-pain expression is neutral expression.

Motivated by this, we propose a regression-based algorithm to address this one-class transfer learning problem. Using the training data of one available class, we use a regressor to predict the other unknown class. Unlike the conventional imputation approach where a regressor predicts data samples, our regressor intends to predict the distributions of one class from another. The general assumption of transfer learning [1] is that the target and source data are different but somehow related. For example, they can share the model parameters [2] or part of the training data [3, 7]. In our algorithm, the basic assumption is that the relationship between the positive and negative distributions is shared by the target domain and the source domain.

The main contributions of this paper are as follows:

⋄ We identify the novel problem of transfer learning using one-class, rather than two-class, data in the target domain. This has not been addressed before, but exists in many real-world applications.

⋄ We propose a regression-based algorithm to address this problem. Because the success of our TL algorithm depends on both the classifier and the regressor, we propose a new approach to select the most transferable features, which are not only discriminative, but also favorable to the regressor.

⋄ We design new application scenarios where the target domain performance can be improved using the readily available one-class data, such as the non-pain expression in the beginning of face videos, and the initial negative patch in facial landmark detection.

2. Related Work

TL aims to extract the knowledge from one or more source domains and improve learning in the target domain. It has been applied to a wide variety of
applications, such as object recognition [2, 8], sign language recognition [9], and text classification [10].

We denote the source domain data as $D_S = \{(x_{S,1}, y_{S,1}), \ldots, (x_{S,N_S}, y_{S,N_S})\}$ and the target domain data as $D_T = \{(x_{T,1}, y_{T,1}), \ldots, (x_{T,N_T}, y_{T,N_T})\}$, where $x \in \mathcal{X}$ is in the feature space and $y \in \{-1, +1\}$ is the binary label. Given these labels, the target and source data can be divided into positive and negative data respectively, i.e., $D_S = \{D_{S, -}, D_{S, +}\}$ and $D_T = \{D_{T, -}, D_{T, +}\}$.

The conventional TL algorithm can be categorized into three settings [1]. In inductive TL [3, 2, 11], both the source data $D_S$ and the target data $D_T$ are available. The goal is to learn the target classifier $f_T : x_T \rightarrow y_T$. However, when the size of target training data $D_T$ is very small, i.e., $N_T \ll N_S$, learning $f_T$ solely from $D_T$ may suffer serious overfitting problems. TL remedies this problem by using knowledge from the source data $D_S$. TrAdaBoost [3] attempts to utilize the “good” source data, which is similar to the target data, to improve the target Adaboost classifier. Yao et al. [2] extend TrAdaBoost to cases where abundant training data is available for multiple sources. They propose a mechanism to select the weak classifiers from the source that appears to be most closely related to the target. Kulis et al. [8] propose a domain adaption approach for object recognition. From the labeled object categories, they learn a non-linear transformation for transferring the data points from the source to the target domain. Chen et al. [6] propose to use inductive TL to learn a person-specific model for facial expression recognition. In this paper, we learn a person-specific model using only one-class data (negative data). Inductive TL cannot be applied in this setting directly.

In transductive TL [7, 12, 13, 14], the source and target data are available, but only the source data has labels. Transductive TL utilizes the unlabeled target data to “shift” or “adapt” the model in the source domain to the target domain. In the literature, transductive TL is closely related to dataset shift [15, 13], importance reweighting [16, 17, 18, 7, 12] and domain adaptation [19, 20]. Because the classifier $f_T$ cannot be learned directly from the unlabeled target data, a common approach is to shift or reweight the labeled source data, from which a target classifier can be learned. Zadrozny [7] proposes to estimate the source and target marginal distribution $P_S(x_S)$, $P_T(x_T)$ independently and uses the probability ratio $P_T(x_T)$ to reweight the source data. Huang et al. [12] and Sugiyama et al. [13] propose different algorithms to estimate this weight directly. The learning bound of this importance weighting approach is analyzed by Cortes et al. [16]. In the computer vision community, Gopalan et al. [20] propose to learn a domain shift from the source subspace to the target subspace in Grassmann manifold, and project the labeled source data to a subspace close to the target domain. Another approach
for transductive TL is to incorporate the unlabeled target data of the source domain into the training. Si et al. [14] propose to use the unlabeled target data as a regularization term in the discriminative subspace learning, so that the learned subspace can generalize to the target domain.

Finally, the unsupervised TL, such as [21], is applied to an unsupervised learning task, such as clustering or dimensionality reduction, when both the target label and the source label are not available.

This paper studies a new setting of TL, where only one-class data, \( D_{-T} \) or \( D_{+T} \), is available in the target domain, but two-class data, \( D_{-S} \) and \( D_{+S} \), are available in the source domain. To the best of our knowledge, this one-class TL problem has not been addressed in the literature. It is related to but different from the following topics:

- In transductive TL the target data is unlabeled but includes both positive and negative data, whereas in one-class TL the target data is extremely unbalanced, i.e., either positive or negative data is available.

- Similarly, semi-supervised learning utilizes a small amount of labeled data and a large amount of unlabeled data, which includes both positive and negative data.

- One-class SVM [22] focuses on a one-to-many classification problem where we only have the training data of one target class. One-class SVM attempts to learn a tight hyper-sphere to include most target examples. In our one-class transfer learning, we focus on a binary classification problem in the target domain. Although only one-class data is available in the target domain, both classes are available in the source domain. Furthermore, unlike TL, one-class SVM does not consider the difference between the source and target domains.

- Similar to one-class SVM, PU-learning [23] or partially supervised learning [24] only has one-class (positive) labeled data. However, it also needs a large set of unlabeled data, from which the reliable negative samples can be selected and utilized in learning.
3. One-Class Transfer Learning

Typically TL algorithms start with a base classifier learned from the source domain data $D_S$. This base classifier is then updated to a target classifier with target data. For example, in [3, 2] the target boosted classifier is adapted from weak classifiers learned from the source data. In [25, 11], the target classifier is adapted from existing SVMs from source data. The Adaboost classifier has been very popular in the vision community due to its simplicity and power to generalize well. For these reasons, we choose the Adaboost classifier [26] as our base classifier.

3.1. Learning the Base Classifier from Source Data

Adaboost produces a strong classifier by combining multiple weak classifiers, such as trees or simple stumps [27]. Considering that a weak classifier will be updated with the distribution of the target data, we designed a specific form of weak classifier, which solely depends on the distribution of the positive and negative data. Here, a input data vector $x = (x_1, x_2, ..., x_F)^T$ is composed of $F$ features, and we model the distribution of each feature as a Gaussian distribution. The probability density function (PDF) of the $f$th feature $x_f$ is

$$p_f(x) = p(x_f; \mu_f, \sigma_f) = \frac{1}{\sigma_f \sqrt{2\pi}} \exp \left\{ - \frac{(x_f - \mu_f)^2}{2\sigma_f^2} \right\}. \quad (1)$$

The weak classifier of the $f$th feature is shown in Equation 2 of Algorithm 1.

In the source domain, since data from two classes are available, we can directly learn the Adaboost classifier as shown in Algorithm 1. Please note that each weak classifier is associated with a feature. Hence, a byproduct of the classifier learning is a set of the most discriminative feature with minimal error $\{f^{(k)}\}_{k=1..K}$.

As the first attempt to address the one-class TL problem, we use a uni-modal Gaussian for simplicity, and use the feature tuning approach in Section 4.1.1 to convert a multi-modal distribution to an uni-modal distribution. This method works well in our experiments. More complex data can be approximated by a mixture of Gaussians but with the cost of increased model complexity.

---

2Some transductive TL algorithms [7, 19] focus on the transfer of marginal distribution $p(x)$. These algorithms are based on generative models without using a base classifier.

3Because the PDF of a Gaussian distribution is determined by its parameters $\mu$ and $\sigma$, we use its parameters to denote this PDF for simplicity. For instance, the PDFs of positive and negative data distributions of the $f$th feature are denoted as $p_f^+ = (\mu_f^+, \sigma_f^+)$ and $p_f^- = (\mu_f^-, \sigma_f^-)$. 

6
Algorithm 1 Adaboost classifier learning from the source data.

input: Source data $D_S = \{(x_1, y_1), \ldots, (x_N, y_N)\}$, where $x \in \mathbb{R}^F$ and $y \in \{-1, +1\}$.
output: The classifier $y = H(x)$.

Initialize the weights $w^{(1)}_1, \ldots, w^{(1)}_N = \frac{1}{N}$.

for $k = 1$ to $K$ do  
  $\triangleright$ $K$ is the number of weak classifiers.
  for $f = 1$ to $F$ do  
    $\triangleright$ $F$ is the number of features.
    Estimate the distributions of positive and negative classes for the $f$th feature $\{p^+_f, p^-_f\}$.
    Specify the weak classifier as:
    $$h_f(x) = \text{sign} \left[ \log \frac{p^+_f(x)}{p^-_f(x)} \right].$$
    Compute the weighted error: $\varepsilon_f^{(k)} = \sum_{i=1}^{N} w_i^{(k)} I(h_f(x_i) \neq y_i)$, where $I(h_f(x_i) \neq y_i)$ is the indicator function, which equals 1 when $h_f(x_i) \neq y_i$ and 0 otherwise.
  end for
  Find the most discriminative feature with the minimal error: $f^{(k)} = \arg \min_f \varepsilon_f^{(k)}$.
  Set $\alpha^{(k)} = \frac{1}{2} \ln \left( \frac{1 - \varepsilon_f^{(k)}}{\varepsilon_f^{(k)}} \right)$.
  Update the weights: $w_i^{(k+1)} = w_i^{(k)} \exp \{\alpha^{(k)} I(h_{f^{(k)}}(x_i) \neq y_i)\}$. 
end for

return $H(x) = \text{sign} \left[ \sum_k \alpha^{(k)} h_{f^{(k)}}(x) \right]$. 

3.2. One-Class Transfer Learning from Target Data

In the above section, we learn the AdaBoost classifier from the positive and negative distributions of the source domain data. This classifier consists of the selected discriminative features and their weights $\{f^{(k)}, \alpha^{(k)}\}$ and the distributions of these features $\{p^+_f, p^-_f\}$. In the TL setting interested to us, the main objective is to update the base classifier given the one-class data from the target domain. One intuitive approach to achieve this objective is to only update the distributions of the selected features based on the target data, while maintaining the feature selection and their associated weights. Since only one-class target data is available, we employ a regressor to predict the distribution of the other class in the target domain. In order to learn this regressor, we assume that the source data can be divided into multiple sources $D_S = \{D_1, \ldots, D_M\}$, e.g., the training data of facial expression recognition is from multiple subjects.

Algorithm 2 summarizes the regression-based method to update the model with only negative data $D^-_T$ in the target domain. The transfer learning with positive data is the same by switching the label. Figure 2 depicts the diagram of this one-class transfer learning.
Algorithm 2 One-Class (Negative) Data Transfer Learning

**input:** Data of $M$ sources $D_1, ..., D_M$, where $D_m = \{(x_{m,1}, y_{m,1}), ..., (x_{m,N_m}, y_{m,N_m})\}$. The target negative data $D_{T}^-$. The discriminative features and their weights $\{f^{(k)}, \alpha^{(k)}\}_{k=1..K}$.

**output:** The classifier for the target domain $y = H_T(x)$.

for $m = 1$ to $M$ do
  for $k = 1$ to $K$ do
    Estimate the distributions $\{p_{m,f^{(k)}}^-, p_{m,f^{(k)}}^+\}$ of the feature $f^{(k)}$ using $D_m$.
  end for
end for

Given the distributions of $M$ sources, learn the regressors: $p_{f^{(k)}}^+ = R_{f^{(k)}}(p_{f^{(k)}}^-)$.

for $k = 1$ to $K$ do
  Estimate the negative distribution $p_{T,f^{(k)}}^-$ from $D_{T}^-$.  
  Predict the positive distribution $\hat{p}_{T,f^{(k)}}^+ = R_{f^{(k)}}(p_{T,f^{(k)}}^-)$.
  Specify the weak classifier as: $h_{T,f^{(k)}}(x) = \text{sign} \left[ \log \frac{\hat{p}_{T,f^{(k)}}^+(x)}{p_{T,f^{(k)}}^-(x)} \right]$.
end for

return $H_T(x) = \text{sign} \left[ \sum_k \alpha^{(k)} h_{T,f^{(k)}}(x) \right]$.

---

**Figure 2:** Diagram of one-class transfer learning.

Algorithm 2 is composed of two steps. The first step estimates the positive and negative distributions $\{p_{m}^-, p_{m}^+\}_{m=1..M}$ of $M$ sources, which are then used
as the training data to learn the regressor $R$ between the positive and negative distributions\textsuperscript{4}, $\hat{p}^+ = R(p^-)$, with a Gaussian Process Regression (GPR) \cite{28}.

GPR is a non-parametric regressor and has proven its effectiveness in a wide range of applications, such as gaze estimation \cite{29} and object categorization \cite{30}. Here, we assume a noisy observation model $p^+_m = g(p^-_m) + \epsilon_m$, where each $p^+_m$ is a function of $g(p^-_m)$ perturbed by a noise term $\epsilon_m = \mathcal{N}(0, \sigma^2)$. We set the noise variance $\sigma^2$ as the variance of $p^+$ in the training data, and $g(p^-_m)$ is assumed to be a Gaussian process with a covariance function:

$$k(p^-_m, p^-_l) = \exp(-\|p^-_m - p^-_l\|^2). \quad (3)$$

With this assumption, given the training data $\{p^-_m, p^+_m\}_{m=1..M}$ and a new $p^-$, the distribution of $p^+$ can be derived as a Gaussian distribution, and we use its mean as the regression output:

$$\hat{p}^+ = R(p^-) = \hat{k}^T(K + S)^{-1}g, \quad (4)$$

where $K$ and $S$ are $M \times M$ matrices whose entries are $k(p^-_m, p^-_l)$ and $\sigma^2 \delta_{ml}$ respectively, and $k$ and $g$ are $M$-dimensional vectors whose entries are $k(p^-, p^-_m)$ and $p^+_m$ respectively. Here $m$ and $l$ are both matrix indexes from 1 to $M$. In the training of this non-parametric GPR, we only need to estimate the covariance matrix $K$ from the training data.

In the second step, we estimate the distribution $p^-_T$ from the negative target data, and predict the positive distribution $\hat{p}^+_T$ based on Equation 4. Finally, the weak classifiers are updated using $p^-_T$ and $\hat{p}^+_T$.

Notice that we are still using the discriminative features learned from the training data (Algorithm 1), and only update the distributions of selected features. In the next section, we will take one step further and update this feature set with the novel transferable features.

### 3.3. Learning the Transferable Features

In Algorithm 1, the discriminative features are selected based on the positive and negative distributions of the source data. In contrast, for Algorithm 2, the positive distribution of the target domain is predicted through a set of regressors.

\textsuperscript{4}We estimate one regressor for each feature $f$. Subscript $f$ is ignored for simplicity. Because the variance estimation is not robust given the limited number of the target data, we only learn the regressor from the negative mean to the positive mean: $\mu^+ = R(\mu^-)$. We assume the variances of the target and source data are the same, which are estimated from all the source data.
Algorithm 3 Boosting the transferable features

**Input:** Data of $M$ sources $D_1, \ldots, D_M$, where $D_m = \{(x_{m,1}, y_{m,1}), \ldots, (x_{m,N_m}, y_{m,N_m})\}$.

**Output:** Transferable features and their weights $\{\alpha^{(k)}, f^{(k)}\}_{k=1..K}$. 

**Step 1. Predict the positive distributions of $M$ sources**

for $f = 1$ to $F$ do

for $m = 1$ to $M$ do

Estimate the distributions $\{p^-_{m,f}, p^+_{m,f}\}$ of the $f$th feature using $D_m$.

end for

for $m = 1$ to $M$ do

Learn a regressor $p^+ = R_{m,f}(p^-)$ using other sources $\{p^-_{l,f}, p^+_{l,f}\}_{l=1,...,m-1,m+1,...,M}$.

Predict the positive distribution $\hat{p}^+_m = R_{m,f}(p^-_m)$.

end for

end for

**Step 2. Boosting transferable discriminative features**

Initialize the weights $w^{(1)}_1, \ldots, w^{(1)}_M$ of $M$ sources, where $w^{(1)}_m = (w_{m,1}, \ldots, w_{m,N_m})^T$.

for $k = 1$ to $K$ do

for $f = 1$ to $F$ do

Specify the weak classifier as: $h_{m,f}(x) = \text{sign}\left(\log \frac{\hat{p}^+_{m,f}(x)}{p^-_{m,f}(x)}\right)$.

Compute the weighted error: $\varepsilon_f^{(k)} = \sum_{m=1}^M \sum_{i=1}^{N_m} w_{m,i}^{(k)} I(h_{m,f}(x_{m,i}) \neq y_{m,i})$. 

end for

Find the feature $f^{(k)}$ that minimizes the weighted error: $f^{(k)} = \arg \min_f \varepsilon_f^{(k)}$.

Set $\alpha^{(k)} = \frac{1}{2} \ln \left[\frac{1 - \varepsilon_f^{(k)}}{\varepsilon_f^{(k)}}\right]$. 

Update the weights: $w_{m,i}^{(k+1)} = w_{m,i}^{(k)} \exp\{\alpha^{(k)} I(h_{m,f}(x_{m,i}) \neq y_{m,i})\}$. 

end for

return $\{\alpha^{(k)}, f^{(k)}\}_{k=1..K}$.

Since the true positive distribution of the target data may be different from the predicted one, these features can be less than optimal for the classification task in the target domain. Hence, to remedy this issue, we propose a new algorithm to select the transferable features which are especially designed for the one-class transfer learning setting (Algorithm 3).

Algorithm 3 consists of two steps. In the first step, for each source domain, we estimate the negative distribution from data and predict the positive distribution using a regressor trained from other $M-1$ source domains. This leave-one-source-out regressor actually simulates the regression step to be performed in the target domain during transfer learning. We repeat it for $M$ sources to obtain the negative distributions and the predicted positive distributions $\{p^-_m, \hat{p}^+_m\}_{m=1..M}$.

The second step is similar to the discriminative feature selection in Algorithm 1. However, we use the predicted positive distribution $\hat{p}^+_m$, rather than the
true positive distribution $p_{m}^+$, to learn the weak classifier. Compared to the discriminative feature, this step is consistent with our negative transfer learning which updates the weak classifiers based on the predicted positive distribution. Thus, the selected features are expected to be more suitable for the transfer learning task.

Please note that after Algorithm 3 outputs the selected transferrable features and their weights $\{\alpha(k), f(k)\}_{k=1..K}$, we use Algorithm 2 to train the target model. The whole transfer learning procedure with transferrable features is shown in Figure 3. Comparing Figure 3 with Figure 2, the discriminative features are replaced with the transferrable features.

4. Experiments

In this section, we demonstrate the efficacy of our transfer learning algorithms in two applications: pain expression recognition and facial landmark detection.

4.1. Pain Expression Recognition

Previous approaches [31, 32] have shown that a person-specific model significantly outperforms a generic model when adequate person-specific data are available. However, for pain recognition, person-specific positive data is difficult to collect unless some severe conditions induce pain.
Our one-class transfer learning only needs a few negative samples to train the target model. We use the UNBC-McMaster Shoulder Pain Expression Archive database [33] for experiments. This database contains the spontaneous pain expression of 25 subjects with shoulder injuries during their shoulder movement. It includes 203 video sequences (totally 48,398 frames). Each frame is labeled with a pain intensity (PI) from 0 to 16. The frames with PI > 0 are labeled as positive data, and the rest frames are labeled as negative data.

4.1.1. Feature Extraction

Local Binary Pattern (LBP) is used as the facial image feature because of its efficiency and effectiveness in facial expression recognition [34]. Following the method in [35], we first use the eye locations to crop and warp the face region to a $128 \times 128$ image. This face image is divided into $8 \times 8$ blocks. For each block, we extract a LBP histogram with 59 bins (please refer to [35] for details). Finally, the LBP histograms from image blocks are concatenated into a spatially enhanced LBP feature with $59 \times 8 \times 8 = 3776$ dimensions.

Notice that our weak classifier design assumes the uni-modal Gaussian dis-
tribution of the positive and negative data. In order to handle the multi-modal distribution in real-world data, we apply the feature “tuning” approach [36] to the LBP features. This tuning step maps the feature value to the likelihood ratio of positive versus negative: 

\[
L(x) = \log \frac{\max \{p^+(x), \delta\}}{\max \{p^-(x), \delta\}},
\]

where \(x\) is the original feature value, \(L(x)\) is the tuned feature value, \(p^+(x)\) and \(p^-(x)\) are the positive and negative distributions, and \(\delta\) is a small value. We tune each dimension of the LBP feature independently. After feature tuning, the positive and negative data follow two separable uni-modal distributions, as shown in Figure 4.

4.1.2. Pain Recognition Results

Similar to [33], we perform a leave-one-subject-out cross evaluation on 25 subjects, i.e., iteratively taking one subject as the target data for testing and the remaining 24 subjects as the source data for training.
For our one-class transfer learning, the testing process on the target subject is shown in Figure 1. To simulate the real-world application, we test on each video sequence separately. For each sequence, we assume the first few frames are non-pain expression and use those frames as negative target data. This assumption works well in real-world expression recognition systems because in real life people exhibit non-pain expression most of the time, and usually have non-pain expression when starting to use the system. In the pain database, there are 194 out of 203 sequences starting with at least 15 neutral frames. In our test, we divide each sequence into two halves. The negative frames from the first half are used for transfer learning, and we test on the second half. The number of frames for each sequence varies from 66 to 683, with an average of 238.

The first experiment compares the performance of five algorithms:

- **GenericModel** is a baseline approach using the source data to train a generic Adaboost classifier (Algorithm 1);
- **TransModel** is learned using the transfer learning algorithm as described in Algorithm 2. It shares the same feature set as **GenericModel**, but its positive and negative distributions are updated with the target data;
- **FeatureTransModel** uses Algorithm 3 to select the transferable features, and Algorithm 2 to update the distributions;
- **InductTransModel** is an state-of-the-art inductive transfer learning algorithm [2] utilizing both positive and negative data in the target domain. We use the parameter transfer learning as described in [2].
- **DataTransModel** is a naive data transfer learning algorithm that directly combines the negative target data with all positive source data and learn an Adaboost classifier. This algorithm uses the same data as **TransModel** and **FeatureTransModel** use.

All five classifiers use the same number (400) of weak classifiers. The ROC curves of the above algorithms are shown in Figure 5. The area under ROC curves (AUC) is 0.771 for **GenericModel**, 0.795 for **TransModel**, 0.820 for **FeatureTransModel**, 0.895 for **InductTransModel**, and 0.753 for **DataTransModel**.

First, we notice that **DataTransModel** is even worse than the generic model, because we only update the negative data distribution using the negative target data, without considering the transfer of the positive data distribution. Second, we can see that the one-class transfer learning can improve the generic model.
Table 1: AUC for one-class transfer learning using different number of negative data samples $N$.

<table>
<thead>
<tr>
<th>$N$</th>
<th>5</th>
<th>10</th>
<th>15</th>
<th>quarter</th>
<th>half</th>
</tr>
</thead>
<tbody>
<tr>
<td>TransModel</td>
<td>0.756±0.007</td>
<td>0.776±0.006</td>
<td>0.783±0.006</td>
<td>0.786±0.006</td>
<td>0.795±0.006</td>
</tr>
<tr>
<td>FeatureTransModel</td>
<td>0.773±0.007</td>
<td>0.799±0.006</td>
<td>0.802±0.006</td>
<td>0.804±0.006</td>
<td>0.820±0.006</td>
</tr>
<tr>
<td>Generic Model</td>
<td>0.771±0.007</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

With the discriminative feature, transfer learning improves the baseline slightly, but with the selected transferable features, the AUC is improved significantly from 0.771 to 0.82. The state-of-the-art pain recognition system [37] achieved AUC=0.751 using appearance features, and achieved AUC = 0.839 by combining two difference normalized appearance features and a shape feature. Compared to [37], our algorithm needs neutral expression for transfer learning, and can achieve comparable result by only using appearance features.

Note that it is not a fair comparison between InductTransModel and one-class transfer learning, because the former uses both positive and negative target data. Hence, it may be viewed as a loose upper-bound of our methods. There are two inductive transfer learning algorithms, i.e., instance-transfer and parameter-transfer in [2]. Both of them select weak classifiers based on the error rate in the target data. If only negative target data is available, they tend to select weak classifiers to classify all the data as negative, so that the classification error on training data is zero, but the error on the testing data is very large. When $N =$ half, the AUC is 0.54 for InductTransModel with negative target data only.

Although the negative data samples are usually easy to collect, we would like to use as fewer data samples as possible in practical applications. To evaluate the effect of the transfer learning data size, we select the first $N$ negative frames from the testing sequence $N = 5, 10, 15, quarter, half$. For quarter and half, we use all the negative frames from the first quarter or the half of the testing sequence for transfer learning. Table 1 shows the AUC of the transfer learning algorithms using different numbers of negative data samples $N$. Compared to GenericModel, FeatureTransModel can improve the AUC from 0.771 to 0.802 with the first 15 negative frames, which is less than 1 second of the video clip.

Our transfer learning is very efficient, since it only needs to compute the mean

\[ \sigma = \sqrt{AUC(1-AUC)} \] [38], where $n^+, n^−$ are the number of positive and negative testing data samples.
of the target data. It runs in real time (< 30 ms) on a PC with 3.2 GHz CPU. For algorithm training, the transferable feature selection is time consuming (~25 min), but it is only slightly slower than the generic model training (~21.5 min).

4.1.3. Comparison of Two Feature Sets

As we discussed in Section 3.3, transferable feature selection is optimal for our regression-based transfer learning. To demonstrate the efficacy of transferable features, we compare them with the discriminative features (Algorithm 1) regarding their classification and regression accuracy. The top 1, 5, 10, 20, 50, 100, 200 and 400 features selected by two different methods are compared. To evaluate the regression accuracy we compare the predicted positive mean ($\hat{\mu}^+$) and the true mean ($\mu^+$) of the positive data. The average regression error for top $N$ features is:

$$\frac{1}{N} \sum_{i=1}^{N} |\hat{\mu}^+_i - \mu^+_i|.$$ 

The regression and classification errors are shown in Figure 6.

We observe that the top transferable features are better than discriminative features in both classification and regression tasks. We also observe that the top selected features have larger regression errors than the features selected later. That is because a feature is selected based on its classification ability. The top features tend to have both larger data variances and larger distances between positive and negative data. The large regression error may be due to the large data variance. Considering the variance of the positive and negative data, we use the metric $\Delta D$ to measure the regression performance:

$$\Delta D = \frac{1}{N} \sum_{i=1}^{N} \left( \frac{|\hat{\mu}^{i,+} - \mu^{i,+}|}{\sigma^{i,+}} - \frac{|\hat{\mu}^{i,-} - \mu^{i,-}|}{\sigma^{i,-}} \right),$$

where $\mu^{i,+}$ and $\sigma^{i,+}$ are the mean and standard deviation of the positive and negative distribution respectively. This metric is the average difference between the normalized distance from the positive mean and the normalized distance from the negative mean. $\Delta D$ gets smaller when $\hat{\mu}^{i,+}$ is closer to $\mu^{i,+}$ and further from $\mu^{i,-}$. As shown in Figure 6, the top transferable features have much smaller $\Delta D$.

---

6Positive data is only used for this evaluation. It is not available in the test run.
compared to discriminative features. Please notice that $\Delta D$ is negative because $\hat{\mu}_i^+$ is always closer to $\mu_i^+$ than to $\mu_i^-$. This means that the predicted positive mean is not only close to the true positive mean but also far from the true negative mean. This also explains why these transferable features yield better classification results.

4.2. Facial Landmark Detection

Face alignment aims to estimate the location of a set of facial landmarks (e.g., eye corner, mouth corner) on a given image by using a supervisely learned model [39, 40]. One popular model is the Active Shape Model [41], which employs a set of local appearance models (LAMs) to localize each landmark independently, and then uses a global shape model to constrain the solution. Taking the eye corner as an example, during the training procedure, its LAM is discriminatively learned from the positive data (patch extracted from labeled ground-truth location) and the negative data (patch extracted from the neighboring locations). During the testing process, given an initial location, LAM will be applied to all candidate locations within a neighborhood, and the maximum classifier score will determine the estimated eye corner location. Since the LAM is critical to the alignment performance, we apply transfer learning to improve its performance. We view the generic training data as the source domain and the given test image as the target domain. Hence, the local patches around the initial location of the test image are negative samples of the target domain.

We use the Labeled Face Parts in the Wild (LFPW) database [42], which includes images under a wide range of head poses, light conditions, and image qualities. A total of 35 fiducial landmarks are manually labeled in each face image. Our experiment only focuses on the LAM of the left eye corner, but the algorithm can be applied to other landmarks as well.

Figure 7 shows example images from LFPW, along with the labeled left eye corner. As the eye corner appearance varies substantially across different images, a generic eye corner LAM may not work well for an arbitrary unseen face image. In contrast, adapting the generic LAM using the specific image characteristics embodied in the negative data might result in a better LAM for this particular test image.

We randomly split 1135 images from LFPW into 200 training images and 935 testing images. First, all the images are rotated and scaled based on the labeled eye positions. Since each image is labeled four times by different labelers, we extract four $15 \times 15$ patches as the positive samples and randomly select 7 negative positions around ground-truth to extract negative samples. Each $15 \times 15$ patch is
reshaped to a 115-dimensional vector and tuned using the same feature tuning method as described in Section 4.1.1. Similarly, we can extract the positive and negative data from testing images.

First, we train a generic eye corner LAM from training data using Algorithm 1. However, this classifier performs poorly on the testing data (AUC = 0.613 ± 0.011). Since this classifier works well on the training data (AUC = 0.940 ± 0.008), this poor testing result is contributed by the large variation between the training and testing data. To address this problem, in each testing image, we extract 5 negative examples around the initial landmark location, and use our one-class transfer learning algorithm to update the classifier. This updated classifier improves the AUC to 0.665 ± 0.011.

To test our classifiers for eye corner detection, we start from an initial eye corner position, which is detected by the PittPatt face-eye detector\(^7\), and search a

\(^7\)http://www.pittpatt.com/
neighborhood around the initial position to find the maximal classifier output as our detection result. For our transfer learning, we randomly extract a few negative examples around the initial position to update the classifier. An example of the detection result and the classifier output score map of two LAMs are shown in Figure 8. The red circle is the ground truth. The red cross is the detection result. The green cross and stars represent the initial position and positions to extract negative transfer learning data. The results show that the generic LAM fails because of the appearance of a large eye shadow. By updating the model with examples from a small neighborhood, our transfer learning can improve the classification result and more importantly, results in a score map with fewer high scores (fewer bright pixels), which indicates the improved detection reliability. In our experiment, we randomly select an initial position which is up to $0.15d$ ($d$ is the interocular distance) from the ground truth. The average detection error over 935 testing images is $0.0919d$ for the generic LAM and $0.0834d$ for the transfer learning LAM. Although the improvement may appear to be small, it is significant ($t = 4.48, p < 0.05$ in $t$-test). Here, no comparison is performed between our one-landmark detection result and the state-of-the-art face alignment algorithm (e.g., [42]), since [42] detects 35 landmarks jointly with the help of a global face shape model.

5. Conclusions

This work identified a new problem of transfer learning, where only one-class data is available. This problem is not uncommon in real-world applications, but has not been studied before. We introduced a new regression-based one-class transfer learning algorithm to address this problem. In this algorithm, we introduced a new feature selection framework for selecting the transferable features that are not only discriminative between the negative and positive data, but also excellent in predicting the positive data distribution from the negative data. We applied our algorithm to facial expression recognition and facial landmark detection. Compared to the generic model without transfer learning, our algorithm with the transferable features can improve both applications with only a few negative examples. This is the first attempt to address such a one-class transfer learning problem. Our framework is general and applicable to a wide range of learning problems where only one-class target data is available. The main assumption of our algorithm is that multiple sources are required, each consisting of a pair of positive and negative distributions. Some applications, like object recognition in [2], aims at solving a one-to-many classification problem, where multiple sources may
share the same large background. Our algorithm cannot be applied to such a setting. Another limitation is that our algorithm is only applicable to classifiers that are directly derived from data distribution, such as the specific Adaboost classifier described in Section 3.1. Further research is required in order to generalize it to other classifiers such as SVM and kNN.

References


