Proceedings of Machine Learning Research 61:1–14, 2018 Conformal and Probabilistic Prediction and Applications

Transfer learning for the Probabilistic Classification Vector Machine

Christoph Raab CHRISTOPH.RAAB@FHWS.DE
University of Applied Sciences Würzburg-Schweinfurt
Department of Computer Science
Würzburg, Germany

Frank-Michael Schleif SCHLEIFY@CS.BHAM.AC.UK
University of Birmingham
School of Computer Science
Edgbaston B15 2TT Birmingham, United Kingdom

Abstract

Transfer learning is focused on the reuse of supervised learning models in a new context. Prominent applications can be found in robotics, image processing or web mining. In these fields, the learning scenarios are naturally changing but often remain related to each other motivating the reuse of existing supervised models. Current transfer learning methods are not well suited and used for sparse and interpretable models. Sparsity is very desirable if the methods have to be used in technically limited environments and interpretability is getting more critical due to privacy regulations. In this work, we show how transfer learning can be integrated into the sparse and interpretable probabilistic classification vector machine and it is compared with different standard benchmarks in the field.

Keywords: Transfer Learning, Probabilistic Classification Vector Machine, Transfer Kernel Learning

1. Introduction

Supervised learning and particular classification, is an important task in machine learning with a broad range of applications (Bishop, 2009). The obtained models are used to predict the target label of unlabeled test samples. In general it is assumed that the underlying domain of interest is not changing between training and test samples. If the domain is changing from one task to a related but different task one would like to reuse the available learning model. Domain differences are quite common in real-world scenarios and may lead to substantial performance drops (Weiss et al., 2016).

A practical example is the classification of web pages. In one domain the classifier is trained on university web pages with a word distribution according to universities and in the test scenario the domain has changed to non-university web pages where the word distribution may not be similar to the training distribution. One aim of transfer learning is it to solve the divergence in distribution (Pan and Yang, 2010).

Multiple transfer learning methods have been already proposed, following different strategies and solving various problems (Weiss et al., 2016; Pan and Yang, 2010). In this paper we focus on sparse models which are not yet covered sufficiently by transfer learning approaches.

The probabilistic classification vector machine (PCVM) (Chen et al., 2009) is a sparse probabilistic kernel classifier, pruning unused basis functions during training. The PCVM is a very successful classification algorithm (Chen et al., 2009; Schleif et al., 2015) with competitive performance to Support Vector Machine (SVM) (Cortes and Vapnik, 1995), but is additionally natural sparse and creates interpretable models as needed in many application domains of transfer learning. The original PCVM is not well suited for transfer learning.

To tackle this issue, we will extend the probabilistic classification vector machine with transfer learning through integration of a transfer kernel approach and solve the resulting problem of parameter determination through a heuristic estimation. The proposed solution is tested against other commonly used transfer learning approaches and data.

Subsequently we introduce the used algorithmic concepts in Sec 3 followed by an experimental part in Sec 4, addressing the classification performance and the sparsity of the model. A summary and open issues are provided in the conclusion at the end of the paper.

2. Methods

2.1. Probabilistic Classification Vector Learning

Probabilistic Classification Vector Machine (PCVM) (Chen et al., 2009) uses a probabilistic kernel regression model:

$$l(\mathbf{x};\mathbf{w},b) = \Psi\left(\sum_{i=1}^{N} w_i \phi_i(\mathbf{x}) + b\right) = \Psi\left(\Phi(\mathbf{x})^\top \mathbf{w} + b\right) \tag{1}$$

With a link function $\Psi(\cdot)$, with w_i being the weights of the basis functions $\phi_i(\mathbf{x})$ and b as bias term. In PCVM the basis functions ϕ_i are defined explicitly as part of the model design. In (1) the standard kernel trick can be applied. The implementation of PCVM (Chen et al., 2009) use the probit link function, i. e.:

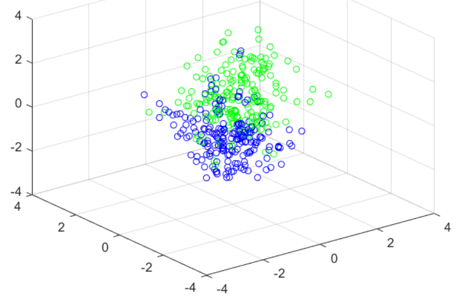
$$\Psi(\mathbf{x}) = \int_{-\infty}^{x} \mathcal{N}(t|0,1)dt \tag{2}$$

Where $\Psi(\mathbf{x})$ is the cumulative distribution of the normal distribution $\mathcal{N}(0,1)$. Chen et al. (2009) use the Expectation-Maximization algorithm for learning the model. The underlying optimization framework within EM, prunes unused basis functions and, therefore, is a sparse probabilistic learning machine. In PCVM we will use the standard RBF-kernel with a Gaussian width θ. In (Schleif et al., 2015) a PCVM with linear costs was suggested, which makes use of the Nyström approximation, a technique also used in the subsequently proposed transfer learning PCVM. Further details can be found in (Chen et al., 2009) and (Schleif et al., 2015)

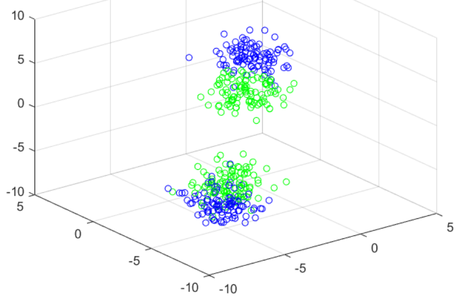
2.2. Transfer learning

Transfer learning is the task of reusing information or trained models in one domain to help to learn a target predictive function in a different domain of interest (Pan and Yang, 2010).

This difference in training or testing data and tasks is formalized as (Weiss et al., 2016): Let $\mathcal{D}$ be a domain, composed by samples x_i with $x_i \in \mathcal{X}^D$, $i \in [1, \ldots, N]$, where N is the number of samples and $\mathcal{X}^D \subseteq \mathcal{F}$ is a D-dimensional feature space. We introduce the short cut $\mathcal{F}_\mathcal{X}$ to refer to the feature space $\mathcal{X}^D$ sampled from $\mathcal{F}$. Further, we consider a marginal probability distribution $P(\mathbf{X})$, using a data matrix $\mathbf{X}$ sampled from $\mathcal{X}^D$. Formally, this means $\mathcal{D} = \{\mathcal{F}, P(\mathbf{X})\}$. In general, if two domains $\mathcal{X}$ and $\mathcal{Z}$ are different, they either have different feature spaces $\mathcal{F}_\mathcal{Z} \neq \mathcal{F}_\mathcal{X}$, called *Heterogeneous Transfer*, or different marginal distributions $P(\mathbf{X}) \neq P(\mathbf{Z})$ referred as *Homogeneous Transfer*. In each of these cases transfer learning is needed to obtain optimal prediction models.
The issue of different distributions is illustrated in Figure 1. In Figure 1a a traditional problem with similar source and target probability distributions is shown. The Figure 1b shows the problem of transfer learning where conditional and marginal probability distributions are different. The blue and green colors are representing two classes. From these classes, training and testing sets are sampled. Moreover, Figure 1b shows the problem with two domains.
A task $\mathcal{T}$ refers to a decision problem involving a given domain $\mathcal{D}$, a label set $\mathcal{Y}$, for example $\mathcal{Y} = \{-1, +1\}$ (binary classification), and a parametric decision function $f(\cdot)$ defined on $\mathcal{D}$ using $\mathcal{Y}$. The number of possible classes is denoted by C. In case of the PCVM it is given by $f(x) = \Phi(\mathbf{x})^\top \mathbf{w} + b$. It is formalized by $\mathcal{T} = \{\mathcal{Y}, f(\cdot)\}$ with $\mathbf{y} \in \mathcal{Y}$. The vector $\mathbf{y}$ represent the (predicted) labels for *all* data points regarding the label set $\mathcal{Y}$. The function $f(\cdot)$ is a predictive function to make predictions for unseen examples, i. e. new data points $\mathbf{x}$. From a probabilistic viewpoint, we can interpret $f(\mathbf{x})$ as $P(y \mid \mathbf{x})$. In general, two tasks $\mathcal{T}_\mathcal{X}$ and $\mathcal{T}_\mathcal{Z}$ are different if they have varying conditional probability distributions or label spaces, which means $\mathcal{Y}_\mathcal{X} \neq \mathcal{Y}_\mathcal{Z} \vee P(y \mid \mathbf{x}) \neq P(y \mid \mathbf{z})$, called task transfer learning.



(a)



(b)

Figure 1: Comparison of distributions for two classes (indicated by blue/green or dark/bright coloring) in the traditional (left) and the transfer learning concept (right). The left shows one domain where the classes are similarly distributed. The right figure shows two different domains with different marginal *and* conditional probability distributions. Axis labeling is arbitrary. Best viewed in color.

2.3. Related Work

Following the aforementioned scenarios of transfer learning, a variety of solutions have been proposed. One may distinguish roughly five approaches, which are trying to solve the homogeneous transfer problem: Instance-transfer, symmetric-feature transfer, asymmetric-feature transfer, relational-knowledge transfer and parameter transfer (Weiss et al., 2016), here we briefly discuss the first three techniques, while the remaining ones are addressed later on in the proposed approach.

The *instance transfer method* tries to align the marginal distribution by re-weighting some source data. This re-weighted data is then directly used with target data for training. It seems that these type of algorithm works best when the conditional probability is the same in source and the target domain and only aligns marginal distribution divergences (Weiss et al., 2016). An example, called TrAdaBoost, is given in Dai et al. (2007b).

Approaches implementing the symmetric feature transfer are trying to find a common latent subspace for source and target domain with the goal to reduce the marginal distribution differences, such that the underlying structure of the data is preserved in the subspace. An example of a symmetric feature space transfer method is the Transfer Component Analysis (TCA) (Pan et al., 2011; Weiss et al., 2016). Another example is Geodesic Flow Kernel (GFK), which finds a particular subspace by projecting original data into a Grassmannian manifold and applying the kernel trick (Gong et al., 2012).

The asymmetric feature transfer learning approach tries to transform the source domain data in the target (subspace) domain. This should be done in a way that the transformed source data will match the target distribution. In comparison to the symmetric feature transfer approaches, there will be no shared subspace, but only the target space. An example is given by the Joint Distribution Adaptation (JDA) algorithm (Long et al., 2015; Weiss et al., 2016).
All the considered methods have approximately a complexity of $\mathcal{O}(N^2)$ where N is the largest number of samples concerning test or training (Long et al., 2013; Pan et al., 2011; Gong et al., 2012).

The algorithms GFK, JDA and TCA doing *transductive* transfer learning (Pan and Yang, 2010), meaning they need *unlabeled* target data at training time. The TrAdaBoost approach needs a very small amount of *labeled* target data and, therefore, is doing *inductive* transfer learning (Pan and Yang, 2010).

The mentioned solutions do not take the label information into account in solving the transfer learning problem, e. g. to find new feature representations. These solutions can not be directly used as predictors, but rather are wrappers for classification algorithms. The baseline classifier is most often the *Support Vector Machine* (SVM).

3. Probabilistic Classification Vector Machine with transfer learning

According to (Chen et al., 2009), the SVM has some drawbacks, mainly a rather dense decision function (also in case of so called sparse SVM techniques) and a lack of a mathematically sound *probabilistic* formulation. The PCVM addressed this issues providing a competitive sparse and probabilistic classification function (Chen et al., 2009). This was the original motivation to expand PCVM by transfer learning.

The proposed transfer learning solution is the Probabilistic Classification Transfer Kernel Vector machine (PCTKVM). It combines the Transfer Kernel Learning concept (TKL) (Long et al., 2015) and the PCVM formulation (Chen et al., 2009) or the respective Nyström approximated version (Schleif et al., 2015).

The main idea in TKL is to approximate the kernel of the training set with the kernel of the test set via the Nyström kernel approximation (Long et al., 2015). We will adopt this concept in PCTKVM.

3.1. Nyström Approximation

The Nyström Method originally aims at solving eigenvalue equation and was first used in machine learning to approximate a kernel matrix (Williams and Seeger, 2001). The approximation is performed by sampling[1] M observations (landmarks) from kernel matrix $\mathbf{K}$. This kernel is obtained by evaluating the respective kernel function for the considered data points. An approximated kernel matrix $\tilde{\mathbf{K}}$ is achieved by solving:

$$\tilde{\mathbf{K}} = \mathbf{K}_{N,M}(\mathbf{K}_{M,M})^{-1}\mathbf{K}_{M,N} \tag{3}$$

With $\mathbf{K}_{N,M}$ being a sub-matrix of $\mathbf{K}$ using all N rows and M landmark columns. The approximation is exact if the original kernel matrix $\mathbf{K}$ has rank M and the samples are linearly independent columns from the kernel matrix.

3.2. Domain Invariant Kernel Learning

TKL calculates the kernel evaluations of the training dataset using a Nyström approximation of the *test* data and the respective kernel expansion to new data points. It can be seen as Relational-Knowledge-Transfer solution. The primary challenge is to find eigen*values* such that the difference between the approximation and the ground truth kernel of the training data is minimal. This forms a domain invariant kernel for transfer learning (Long et al., 2015). Let $\mathbf{Z} = \{\mathbf{z}_1, \ldots, \mathbf{z}_N\}$ be training data, sampled from $p(\mathbf{Z})$ in the training domain $\mathcal{Z}$ and $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_M\}$ a test dataset sampled from $p(\mathbf{X})$ in the test domain $\mathcal{X}$, we obtain the associated kernel matrices $\mathbf{K}_\mathcal{Z}$ for training and $\mathbf{K}_\mathcal{X}$ for testing, using an appropriate kernel function (e. g. the rbf-kernel). For the sake of clarity, we define the kernel over training *and* testing data as:

$$\mathbf{K}_\mathcal{A} = \begin{bmatrix} \mathbf{K}_\mathcal{Z} & \mathbf{K}_{\mathcal{Z}\mathcal{X}} \\ \mathbf{K}_{\mathcal{X}\mathcal{Z}} & \mathbf{K}_\mathcal{X} \end{bmatrix} \tag{4}$$

With $\mathbf{K}_\mathcal{Z}$ for training, $\mathbf{K}_\mathcal{X}$ for testing and $\mathbf{K}_{\mathcal{Z}\mathcal{X}} = \mathbf{K}_{\mathcal{X}\mathcal{Z}}^T$ as *cross-domain* kernel between domains. Given a kernel $\mathbf{K}_\mathcal{X}$, using the eigensystem $\{\lambda_i, \phi_i(\mathbf{x})\}$ the kernel can be evaluated for any new data point $\mathbf{z}$ and finally generating the kernel $\mathbf{K}_\mathcal{Z}$ using a kernel expansion as discussed, e. g. in Williams and Seeger (2001). Again the training kernel is approximated via the Nyström approach (Long et al., 2015):

$$\mathbf{K}_\mathcal{Z} \simeq \mathbf{U}_\mathcal{Z}\mathbf{\Lambda}_\mathcal{X}\mathbf{U}_\mathcal{Z}^T = \mathbf{K}_{\mathcal{Z}\mathcal{X}}\mathbf{K}_\mathcal{X}^{-1}\mathbf{K}_{\mathcal{X}\mathcal{Z}} \tag{5}$$

1. Advanced selection concepts are given in Kumar et al. (2012), however, this is not crucial in our approach.

Where $\mathbf{U}_\mathcal{Z}$ are eigenvectors of the *training* kernel and $\mathbf{\Lambda}_\mathcal{X}$ as eigenvalues of the *test* kernel. Revisiting, the original *Nyström* approximation samples some landmarks from the original kernel to obtain $\mathbf{K}_\mathcal{X}$. The *Transfer Kernel Learning* approach uses all target data as landmarks and, therefore, the number of landmarks for $\mathbf{K}_\mathcal{X}$ is naturally M.
A remaining problem is the difference between test and training feature space probability distributions. This means that if $p(\mathbf{X}) \neq p(\mathbf{Z})$, the error of the approximation from Eq. (5) can be arbitrarily large (Long et al., 2015). The main improvement done by Long et al. is to solve this problem and to learn the domain invariant approximated kernel $\bar{\mathbf{K}}_\mathcal{A}$, by remodeling the eigenvalues based on an error minimization with the assumption that it is sufficient that $\mathbf{K}_\mathcal{Z} \simeq \mathbf{K}_\mathcal{X}$ instead of $p(\mathbf{Z}) \simeq p(\mathbf{X})$ (Long et al., 2015).
However, another potential problem could be the different number of samples in the two domains: $\mathbf{K}_\mathcal{Z} \in \mathbb{R}^{N\times N}$ and $\mathbf{K}_\mathcal{X} \in \mathbb{R}^{M\times M}$ with $N \neq M$. Therefore the approximated (by Long extrapolated) source kernel must be $\bar{\mathbf{K}}_\mathcal{Z} \in \mathbb{R}^{N\times N}$, by using the eigensystem of the target kernel $\mathbf{K}_\mathcal{X}$. The kernel $\bar{\mathbf{K}}_\mathcal{Z}$ will be compared to the actual training kernel $\mathbf{K}_\mathcal{Z}$(Long et al., 2015). To extrapolate the kernel, first, the eigenvectors have to be *extrapolated*: (Long et al., 2015)

$$\bar{\mathbf{U}}_\mathcal{Z} \simeq \mathbf{K}_{\mathcal{Z}\mathcal{X}}\mathbf{U}_\mathcal{X}\mathbf{\Lambda}_\mathcal{X}^{-1}. \tag{6}$$

In the approach of Long et al. (2015), a *relaxation* of the eigenvalues is done by considering the eigenvalues as learnable parameters $\mathbf{\Lambda} = diag[\lambda_1, \ldots, \lambda_M]$. Therefore the kernel extrapolation can be expressed as: (Long et al., 2015) $\bar{\mathbf{K}}_\mathcal{Z} = \bar{\mathbf{U}}_\mathcal{Z}\mathbf{\Lambda}\bar{\mathbf{U}}_\mathcal{Z}^T$. Hence, some $\mathbf{\Lambda}$ does not necessarily reduce the distribution difference and has to be well chosen. This approach should preserve the original structure of the test domain while remaining flexible to solve the difference in the distributions. The optimization problem with respect to $\mathbf{\Lambda}$ is given as (Long et al., 2015):

$$\begin{gathered} \min_{\mathbf{\Lambda}} ||\bar{\mathbf{K}}_\mathcal{Z} - \mathbf{K}_\mathcal{Z}||_F^2 = ||\bar{\mathbf{U}}_\mathcal{Z}\mathbf{\Lambda}\bar{\mathbf{U}}_\mathcal{Z}^T - \mathbf{K}_\mathcal{Z}||_F^2 \\ \lambda_i \geq \zeta\lambda_{i+1}, i = 1, \ldots, M-1 \quad \lambda_i \geq 0, i = 0, \ldots, M \end{gathered} \tag{7}$$

With $\zeta \geq 1$ the eigenspectrum dumping factor. The obtained kernel can be used in any kernel machine. The complexity of the TKL algorithm can be given with $\mathcal{O}((D+R)(N+M)^2)$, where R denotes the number of used eigenvectors, D refers to the dimensionality of data.(Long et al., 2015).

3.3. PCTKVM Algorithm

By using the TKL approach we obtain a new kernel usable in PCVM. In the training of the PCVM, we are using the *extrapolated* training kernel: $\bar{\mathbf{K}}_\mathcal{Z} = \bar{\mathbf{U}}_\mathcal{Z}\mathbf{\Lambda}\bar{\mathbf{U}}_\mathcal{Z}^T$.
In general an rbf-kernel with in-place optimized distribution-width parameter θ is used in PCVM. Accordingly, a simple replacement of the standard rbf-kernel by a rbf-kernel obtained with TKL can be inefficient. In PCVM the kernel is recalculated in each iteration, based on the optimized θ from the previous iteration. Consequently, we would have to recalculate the entire transfer kernel too. The complexity of the standard PCVM is $\mathcal{O}(K^3)$ with K as number of basis functions and $K = N$ at the beginning of the training and before pruning basis functions. The complexity of TKL is $\mathcal{O}((D+R)(N+M)^2)$. Combining the two, we would end up with a computational complexity of $\mathcal{O}(N^3(D+R)(N+M)^2)$.

The performance of the PCTKVM strongly depends on the quality of θ, hence some reasonable θ optimization is needed. A simple optimization, e. g. every ten iterations, was found to be insufficient. Having θ fixed could solve the problem, given a reasonable θ can be found.
To achieve this, we use the approach given in Kitayama and Yamazaki (2011). It is a heuristic based on a distance measure to obtain the width for the kernel. It can determine a good width by iterating and *min-max*-rescaling until some criterion is reached.

The PCVM makes the assumption of zero-mean Gaussian distributed data to ensure that the underlying sparsification approach remains valid. To ensure this, the data are typically normalized by a z-score transformation prior learning. To keep this model assumption, we replaced the *min-max*-scaling with *z*-score scaling in the θ estimation approach. As a consequence, only an initial scaling is necessary and the iterative algorithm of Kitayama and Yamazaki (2011) reduces to a simple minimum search:

$$\theta_i = \frac{d_{i,max}}{\sqrt{D}\sqrt[D]{(N+M)-1}}, i = 1, \dots, (N+M) \tag{8}$$

with D as the dimensionality of the data and $d_{j,max}$ is the maximum (Euclidean)-distance between the i-th data point and each other. Using this simplifications we end up with a complexity of $\mathcal{O}(N^3 + (D+R)(N+M)^2)$. Considering D and R as constant we obtain $\mathcal{O}(N^3+M^2)$ for the PCTKVM with a fixed theta. The PCTKVM is presented in Algorithm 1. Predictions are made using the PCVM predictive function but with the use of $\bar{\mathbf{K}}_{\mathcal{XZ}} = \mathbf{U}_{\mathcal{X}}\mathbf{\Lambda}\bar{\mathbf{U}}_{\mathcal{Z}}^T$ as kernel for test data, pointed out by Long et al. (2015). However, in case of the

Algorithm 1 Probabilistic Classification Transfer Kernel Vector Machine

```
Require: $\mathbf{K} = [\mathbf{Z};\mathbf{X}]$ as $M$ sized training and $N$ sized test set; $\mathbf{Y}$ as $N$ sized training label
    vector; kernel(-type) $ker$; eigen-spectrum dumping factor $\zeta$; $\theta$ as kernel parameter.
Ensure: Weight Vector $\mathbf{w}$; bias $b$, kernel parameter $\theta$; transfer kernel $\bar{\mathbf{K}}_{\mathcal{A}}$.
 1: $\mathbf{D}$ = calculate_dissimilarity_matrix($\mathbf{K}$);
 2: $\theta$ = theta_estimation($\mathbf{D}$);                          ▷ According to eq. (8)
 3: $\bar{\mathbf{K}}_{\mathcal{A}}$ = transfer_kernel_Learning($\mathbf{D}$,$ker$,$\theta$,$\zeta$);      ▷ According to (3.2)
 4: [$\mathbf{w}$,$b$] = pcvm_training($\bar{\mathbf{K}}_{\mathcal{Z}}$);             ▷ According to Chen et al. (2009)
```

SVM as baseline classifier, the kernel with size $N \times M$ is used and restricted to the respective support vectors. The predictive function for the SVM has the form $\mathbf{y} = \bar{\mathbf{K}}_{\mathcal{XZ}}(\alpha \cdot \mathbf{y_Z}) + b$, where α are the Lagrange multipliers (Long et al., 2015).
Because of the sparsity of the PCVM the number of basis functions used in the decision function is typically small[2]. If we consider that our model has S non zero weight vectors with $S \ll N$ and because the PCVM uses only kernel rows/columns corresponding to the non zero weight vector index, then our final kernel $\bar{\mathbf{K}}_{\mathcal{SX}}$ for prediction has size $(S \times M)$. Therefore, the predictive function of the PCTKVM has the form: $\mathbf{y} = \bar{\mathbf{K}}_{\mathcal{SX}}\mathbf{w} + b$. The probabilistic output is calculated with the probit link function used in the PCVM.

2. The decision function may be constructed from e. g. 10 samples only.

4. Experiments

We follow the experimental design which is typical for transfer learning algorithms (Long et al., 2015; Gong et al., 2012; Long et al., 2013; Pan and Yang, 2010). A crucial characteristic of the datasets for transfer learning is that domains for training and testing are different but related. This relation exists because the train and test classes have the same top category or source. The classes itself are subcategories or subsets.

4.1. Benchmark Datasets

The study consists of 18 benchmark datasets, already preprocessed and taken from Gong et al. (2012) and Long et al. (2015).
Three of them are text-based from Reuters-21578[3] and are a collection of Reuters newswire articles collected in 1987. The text is converted to lower case, words are stemmed and stopwords are removed. With the Document Frequency (DF)-Threshold of 3, the numbers of features are cut down. Finally, Term-Frequency Inverse-Document-Frequency (TFIDF) is applied for feature generation (Dai et al., 2007a). The three top categories *organization (orgs)*, *places* and *people* are used in our experiment.
To create a transfer problem, a classifier is not tested with the same categories as it is trained on, i. e. it is trained on some subcategories of organization and people and tested on others. Therefore, six datasets are used: *orgs vs. places*, *orgs vs. people*, *people vs. places*, *places vs. orgs*, *people vs. places* and *places vs. people*. There are two-class problems with the top categories as positive and negative class and with subcategories as training and testing examples. The choice of subcategories is the same as in Long et al. (2015). To reproduce the results below one should use the linked version of the Reuters dataset.
The remaining twelve are images datasets from Caltech-256[4] and Office. The first, Caltech (C) is the largest dataset of images and contains 30607 images within 257 categories. The last is a collection of images drawn from three sources which are from *amazon (A)*, digital SLR camera *dslr* and *webcam (W)*. They vary regarding camera, light situation and size, but having 31 object categories, e. g. computer or printer, in common. Duplicates are removed, as well as images which have more than 15 similar Scale Invariant Feature Transform (SIFT) in common.
To get an overall collection of the four image sets, which are considered as domains, categories with the same description are taken. From the 31 and 256 categories, ten similar categories are extracted: backpack, touring-bike, calculator, head-phones, computer-keyboard, laptop-101, computer-monitor, computer-mouse, coffee-mug and projector. They are the class labels from one to ten.
With this, a classifier should be trained on the training domain, e. g. on projector images (Class One) from amazon (Domain A), and should be able to classify the test image to the corresponding image category, e. g. projector (Class One) images from Caltech (Domain C) against other image types like head-phones (Class Two). The final feature extraction is done with Speeded Up Robust Features Extraction (SURF) and encoded with 800-bin histograms. Finally, the twelve sets are designed to be trained and tested against each other by the ten labels (Gong et al., 2012).

3. http://www.daviddlewis.com/resources/testcollections/reuters21578
4. http://www.vision.caltech.edu/Image_Datasets/Caltech256/

Name	#Examples 1	#Examples 2	#Features	# Labels
Caltech vs. Amazon	1123	958	800	10
Caltech vs. DSLR		295		
Caltech vs. Webcam		157		
Orgs vs. People	1237	1208	4771	2
Orgs vs. Places	1208	1016		
People vs. Places	1016	1208		

Table 1: Overview of the key figures of the Image dataset.

A summary of all datasets is shown in Table 1. Regardless of the dataset, features have been normalized to standard mean and variance. The samples for training and testing the classifiers are drawn with five times two-fold cross-validation suggested by Alpaydm (1999). Furthermore, following a scheme of data sampling adapted to transfer learning as suggested in Gong et al. (2012).

4.2. Details of Implementation

Most of the algorithms used in the comparisons employ the SVM algorithm and the rbf-kernel as mentioned before. The GFK algorithm is an exception and is using the k nearest neighbor classifier. The LibSVM implementation is used for TCA, GFK, JDA and TKL, employing the respective pre-calculated kernel, with $C = 10$. For PCVM the maximal number of iterations is set to 600 as a common choice. PCTKVM[5] and TKL have the eigenvalue dumping factor ξ as a crucial parameter, which was set to 2 for the text datasets and 1.1 for the image datasets. C and ξ are not optimized via grid search and taken from Long et al. (2015).
The remaining parameters are optimized on the training data sets with respect to best performance on it: JDA has two model parameters. First the number of subspace bases k, which is set to 100 and found via grid-search from $k = \{1, 2, 5, 10, 20, ..., 100, 200\}$. The regularization parameter λ is set to 1 for both sets, determined by a grid search ($\lambda = \{0.1, 0.2, 1, 2, 5, ..., 10\}$) (Long et al., 2015).
For the GFK solution the parameter, number of subspace dimensions, is evaluated by a grid-search from $k = \{1, 2, 5, 10, 20, ..., 100, 200\}$ within the training data and finally set for the text sets to 40 and to 50 for the image sets.
The TCA has also one parameter which gives the subspace dimensions and is determined from $\mu = \{1, 2, 5, 10, 20, ..., 100, 200\}$ and finally set to $\mu = 50$ for both datasets.

4.3. Comparison of Performance

Experimental results are shown in Table 2 as mean errors from a 10-fold cross-validation over 18 datasets, which are in total 180 test runs. The standard deviation is shown in brack-

5. Matlab code of PCTKVM and datasets can be obtained from `https://github.com/ChristophRaab/pctkvm.git`

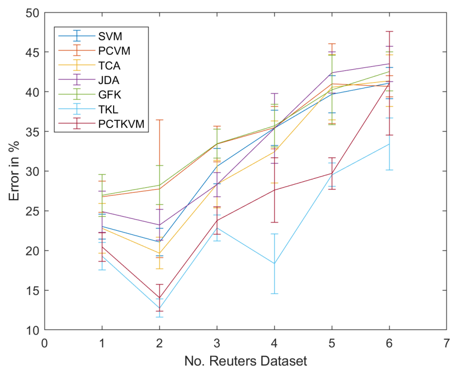
Error Image 2 Domains - 10 Classes	SVM	PCVM	PCTKVM	TCA	JDA	GFK	TKL
C vs A	**52.38** (2.30)	61.77 (2.04)	57.35 (2.88)	53.11 (3.28)	53.32 (3.00)	63.68 (3.49)	53.70 (1.72)
C vs D	57.45 (5.03)	64.33 (5.19)	**56.69** (5.05)	63.44 (4.45)	58.72 (4.31)	65.21 (4.94)	57.71 (4.21)
C vs W	66.70 (4.25)	71.46 (4.21)	63.72 (6.57)	70.30 (5.11)	64.76 (6.34)	73.22 (2.27)	**63.52**(2.35)
A vs C	59.54 (2.61)	65.24 (2.14)	61.91 (2.27)	58.66 (2.12)	58.90 (1.33)	66.20 (1.44)	**58.70**(2.21)
A vs D	66.90 (4.86)	70.86 (5.71)	63.82 (4.09)	64.33 (4.34)	62.33 (6.85)	72.25 (4.08)	**60.90** (5.10)
A vs W	69.51 (1.91)	70.91 (2.59)	66.67 (5.05)	69.44 (4.25)	67.34 (2.42)	76.01 (2.07)	**64.29** (2.94)
D vs C	75.41 (1.13)	78.15 (2.05)	74.07 (3.09)	71.83 (1.78)	72.38 (2.04)	72.47 (2.38)	**70.94** (2.52)
D vs A	73.63 (1.61)	77.06 (2.52)	74.03 (2.47)	**68.93** (2.97)	66.99 (2.68)	71.25 (2.34)	70.39 (2.73)
D vs W	45.77 (2.97)	64.54 (5.35)	47.46 (4.74)	32.81 (3.87)	36.07 (2.84)	36.34 (4.83)	**32.47**(4.55)
W vs C	73.86 (1.52)	76.49 (1.76)	74.14 (2.79)	**68.44** (1.63)	74.07 (1.63)	74.64 (1.39)	68.51 (1.81)
W vs A	71.84 (1.24)	73.07 (1.69)	71.07 (2.92)	**66.28** (1.53)	72.13 (1.55)	71.27 (2.24)	68.08 (1.45)
W vs D	**25.10** (3.60)	43.69 (4.96)	40.76 (3.04)	27.13 (4.61)	25.61 (3.83)	28.29 (4.01)	29.81 (5.20)
RMSE	63.11 (3.06)	68.72 (3.68)	63.47(3.96)	61.23 (3.55)	61.06 (3.67)	65.92 (3.19)	**59.72** (3.32)

Error Image 2 Domains - 2 Classes	SVM	PCVM	PCTKVM	TCA	JDA	GFK	TKL
Orgs vs People	23.01 (1.58)	26.77 (3.18)	20.45 (1.95)	22.78 (3.14)	24.88 (2.61)	26.95 (2.65)	**19.29** (1.73)
People vs Orgs	21.07 (1.72)	27.77 (2.19)	14.05 (8.68)	19.68 (2.00)	23.23 (1.93)	28.23 (2.46)	**12.76** (1.16)
Orgs vs Places	30.62 (2.22)	33.42 (6.10)	23.78 (2.26)	28.38 (3.00)	28.30 (1.51)	33.46 (1.83)	**22.84** (1.62)
Places vs Orgs	35.45 (2.24)	35.49 (8.19)	27.62 (2.66)	32.42 (3.91)	35.37 (4.39)	35.73 (2.69)	**18.33** (3.75)
Places vs People	39.68 (2.35)	41.01 (6.98)	29.71 (5.03)	40.58 (4.11)	42.41 (2.59)	40.24 (4.37)	**29.55** (1.46)
People vs Places	41.08 (1.98)	40.69 (5.52)	41.08 (1.33)	41.39 (3.26)	43.51 (2.23)	42.56 (2.46)	**33.42** (3.28)
RMSE	32.74 (2.03)	34.65 (4.44)	27.43 (3.47)	31.94 (3.31)	33.92 (2.70)	35.00 (2.85)	**23.74** (2.38)

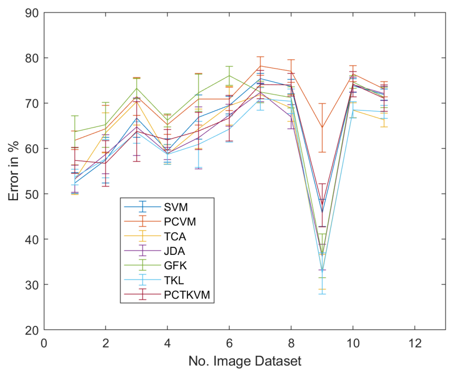
Table 2: Cross-validation comparison of the tested algorithms on 18 domain adaptation datasets by the error and RMSE metrics. Twelve image sets with ten classes each and six text sets with two classes each. Each dataset has two domains. It demonstrates mean of ten runs of cross-validation per dataset with the standard deviation in brackets. The winner is marked with a bold performance value.

ets. The results are shown for Image and Reuters individually. The proposed PCTKVM classifiers is shown in the fourth column. The performance of the best classifier is indicated in bold. In Figure 2, a graph of mean performance and the standard deviation is plotted. In general, and according to the definition of negative transfer from Weiss et al. (2016), it can be seen that negative transfer happens between the baseline SVM and the transfer solution GFK.

We can see that PCTKVM achieves an *overall* better performance result than PCVM in this test. Furthermore, PCTKVM can achieve better performance at text sets than the remaining transfer learning solutions, except TKL, and SVM. The performance of the new PCVM's at image datasets is comparable with the result of SVM, which is also observable in Figure 2b. Nevertheless the other transfer learning solutions are not remarkably better than SVM. Additionally, we can see that the SVM baseliner has an overall better performance than the PCVM baseliner.
It seems that the SVM can handle very small datasets better than the PCVM in this setup. But, again referring to the detailed test results, it states that the PCTKVM is often the second best classifier. The PCTKVM is very competitive to other transfer learning classifiers. The fact that all PCVM's are randomly initialized is not critical because we can observe that the standard deviation is not much higher in comparison with the other classifiers form Table 2.



(*a*)



(*b*)

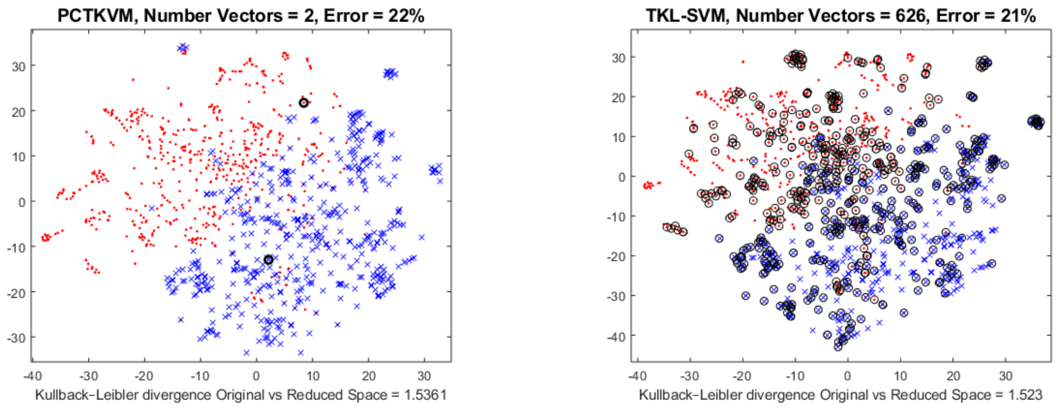
Figure 2: Plot of mean error with standard deviation of the cross-validation test. The left shows the result on Reuters and the right shows the result on images. A graph shows the error and a vertical bar shows the standard deviation. The number (No.) of datasets are the order of datasets in Table 2. Best viewed in color.

4.4. Comparison of Model Complexity

We measured the model complexity by means of the number of model vectors, e. g. support vectors. The result of our experiment is shown in Table 3. We see that the transfer learning models of the PCTKVM provide very sparse models, while having similar good performance as shown in Table 2. The difference in the number of model vectors is significant. The difference in model complexity is exemplary shown in Figure 3. It shows a sample result of classification of PCTKVM and SVM on the text dataset *orgs vs people* with the settings from above, but without bootstrapping. The error value of the first is 22% with two model vectors and for SVM 21% with 626 support vectors.
This clearly demonstrates the strength of the PCTKVM in comparison with SVM based transfer learning solutions. PCTKVM achieves sustain performance by a small model complexity and provides a way to interpret the model. Note that the algorithms are trained in the original feature space and the models are plotted in a reduced space, using the t-distributed stochastic neighbor embedding algorithm (van der Maaten and Hinton, 2008).
In summary, the PCTKVM is a preferable choice, because it is similar to other transfer approaches in terms of prediction performance, but is natural sparse, creates interpretable models and produces reliable probabilistic output. Additionally, the improvement in performance from PCVM to PCTKVM is great, which reaffirms the quality of PCVM as baseline classifier and TKL as transfer-learning wrapper.

N. SV.	SVM	PCVM	PCTKVM	TCA	JDA	TKL
Reuters(1153.66)	482.35	46.93	**3.02**	182.70	220.28	190.73
Image(633.25)	309.90	68.40	*54.88*	253.30	287.433	382.53

Table 3: Average mean of model vectors of a classifier for Reuters and Image datasets. The average number of examples in the datasets are shown on the right side of the title.



(*a*) (*b*)

Figure 3: Sample run on *Orgs vs People* (Text dataset). Red colors for the class *orgs* and blue for the class *people*. This plot includes training and testing data. Model complexity of PCTKVM on the left and SVM on the right. The PCTKVM uses two vectors and achieves an error of 22%. The SVM need 626 vectors and has an error of 21%. The black circled points are used model vectors. Reduced with t-SNE (van der Maaten and Hinton, 2008). Best viewed in color.

5. Conclusions

We successfully integrated a proposed transfer learning algorithm in the PCVM, which forms the PCTKVM. We do so by separating the transfer learning as unsupervised wrapper algorithm and using the PCVM as supervised classification algorithm. In the experiments it is shown that this strategy is efficient within the PCVM framework and the sparsity properties are kept. Furthermore, we solved the problem that the algorithm will be slow through the θ optimization by adopting a simple theta estimation and adjusted it to the needs of the PCVM. Because it uses unlabeled target data, the algorithm is transductive. The proposed solution solves the homogeneous transfer problem, by providing sparse, probabilistic models. Summarizing, we showed the efficiency of PCTKVM in an experiment with 18 datasets. Furthermore, our approach can compete with current transfer learning solutions and is only beaten by one method. To our best knowledge and according to Weiss et al. (2016), PCTKVM is the only sparse and general probabilistic transfer learning method proposed so far. The above facts make the prediction quality of the PCTKVM charming. In future work, it would be of interesting to apply this method in practical applications like robotics and to derive a better θ-estimation.

Acknowledgments

We would like to thank Huanhuan Chen, School of Computer Science and Technology University of Science and Technology of China (USTC) for providing input to the original PCVM.

References

Ethem Alpaydm. Combined 5 × 2 cv F Test for Comparing Supervised Classification Learning Algorithms. *Neural Computation*, 11(8):1885–1892, 1999. ISSN 0899-7667.

Christopher M. Bishop. *Pattern Recognition and Machine Learning // Pattern recognition and machine learning*. Information science and statistics. Springer, New York, NY, corrected at 8. printing 2009 edition, 2009. ISBN 0-387-31073-8.

H. Chen, P. Tino, and X. Yao. Probabilistic Classification Vector Machines. *IEEE Transactions on Neural Networks*, 20(6):901–914, 2009. ISSN 1045-9227.

C. Cortes and V. Vapnik. Support vector network. *Machine Learning*, 20:1–20, 1995.

Wenyuan Dai, Gui-Rong Xue, Qiang Yang, and Yong Yu. Co-clustering based classification for out-of-domain documents. In Pavel Berkhin, Rich Caruana, and Xindong Wu, editors, *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Jose, California, USA, August 12-15, 2007*, pages 210–219. ACM, 2007a. ISBN 978-1-59593-609-7.

Wenyuan Dai, Qiang Yang, Gui-Rong Xue, and Yong Yu. Boosting for transfer learning. In Zoubin Ghahramani, editor, *Machine Learning, Proceedings of the Twenty-Fourth International Conference (ICML 2007), Corvallis, Oregon, USA, June 20-24, 2007*, volume

227 of *ACM International Conference Proceeding Series*, pages 193–200. ACM, 2007b. ISBN 978-1-59593-793-3.

B. Gong, Y. Shi, F. Sha, and K. Grauman. Geodesic flow kernel for unsupervised domain adaptation. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 2066–2073, 2012.

Satoshi Kitayama and Koetsu Yamazaki. Simple estimate of the width in Gaussian kernel with adaptive scaling technique. *Applied Soft Computing*, 11(8):4726 – 4737, 2011. ISSN 1568-4946.

Sanjiv Kumar, Mehryar Mohri, and Ameet Talwalkar. Sampling methods for the Nyström method. *Journal of Machine Learning Research*, 13:981–1006, 2012.

M. Long, J. Wang, G. Ding, J. Sun, and P. S. Yu. Transfer Feature Learning with Joint Distribution Adaptation. In *2013 IEEE International Conference on Computer Vision*, pages 2200–2207, 2013.

M. Long, J. Wang, J. Sun, and P. S. Yu. Domain Invariant Transfer Kernel Learning. *IEEE Transactions on Knowledge and Data Engineering*, 27(6):1519–1532, 2015. ISSN 1041-4347.

S. J. Pan and Q. Yang. A Survey on Transfer Learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1345–1359, 2010. ISSN 1041-4347.

S. J. Pan, I. W. Tsang, J. T. Kwok, and Q. Yang. Domain Adaptation via Transfer Component Analysis. *IEEE Transactions on Neural Networks*, 22(2):199–210, 2011. ISSN 1045-9227.

Frank-Michael Schleif, H. Chen, and Peter Tiño. Incremental probabilistic classification vector machine with linear costs. In *2015 International Joint Conference on Neural Networks, IJCNN 2015, Killarney, Ireland, July 12-17, 2015*, pages 1–8. IEEE, 2015. ISBN 978-1-4799-1960-4.

Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9:2579–2605, 2008.

Karl Weiss, Taghi M. Khoshgoftaar, and DingDing Wang. A survey of transfer learning. *Journal of Big Data*, 3(1):9, 2016. ISSN 2196-1115.

Christopher K. I. Williams and Matthias Seeger. Using the Nyström Method to Speed Up Kernel Machines. In T. K. Leen, T. G. Dietterich, and V. Tresp, editors, *Advances in Neural Information Processing Systems 13*, pages 682–688. MIT Press, 2001.