Domain Adaptation with a Domain Specific Class Means Classifier

Gabriela Csurka(✉), Boris Chidlovskii, and Florent Perronnin

Xerox Research Centre Europe, 6 chemin Maupertuis, 38240 Meylan, France
{Gabriela.Csurka,Boris.Chidlovskii,Florent.Perronnin}@xrce.xerox.com

Abstract. We consider the problem of learning a classifier when we dispose little training data from the target domain but abundant training data from several source domains. We make two contributions to the domain adaptation problem. First we extend the Nearest Class Mean (NCM) classifier by introducing for each class domain-dependent mean parameters as well as domain-specific weights. Second, we propose a generic adaptive semi-supervised metric learning technique that iteratively curates the training set by adding unlabeled samples with high prediction confidence and by removing labeled samples for which the prediction confidence is low. These two complementary techniques are evaluated on two public benchmarks: the ImageClef Domain Adaptation Challenge and the Office-CalTech datasets. Both contributions are shown to yield improvements and to be complementary to each other.

Keywords: Domain adaptation · Self-adative metric learning · NCM

1 Introduction

The shortage of labeled data is a fundamental problem in machine learning applications. While huge amounts of unlabeled data is generated and made available in many domains, the cost of acquiring data labels remains high. Domain adaptation addresses this problem by leveraging labeled data in one or more related domains, often referred as "source" domains, when learning a classifier for unseen data in a "target" domain.

The domains are assumed to be related but not identical and when we apply directly models learned on source domains the performance can be often very poor on the target. This is especially true in computer vision applications as existing image collections used for *e.g.* object categorization present specific characteristics which often prevent a direct cross-dataset generalization. The main reason is that even when the same features are extracted in both domains, the underlying cause of the domain shift (changes in the camera, image resolution, lighting, background, viewpoint, and post-processing) can strongly affect the feature distribution and thus violate the assumptions of the classifier trained on the source domain. Therefore it is important during the learning process to infer models that adapt well to the test data they will be deployed on.

L. Agapito et al. (Eds.): ECCV 2014 Workshops, Part III, LNCS 8927, pp. 32–46, 2015.
DOI: 10.1007/978-3-319-16199-0_3

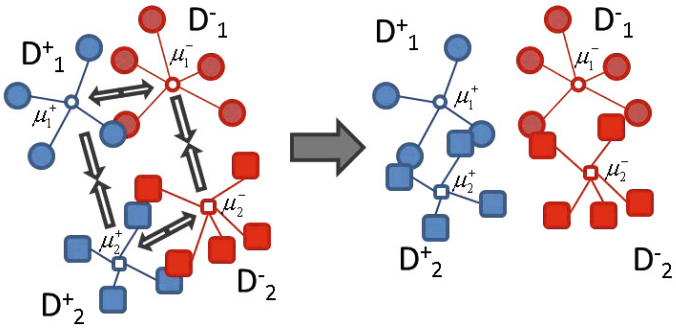

Fig. 1. Exploiting class and domain-related labels to decreased interclass distances and increase intraclass distances independently of the domain

Hence one of the main issues of domain adaptation is how to deal with data sampled from the different distributions and how to compensate this mismatch by making use of information coming from both source and target domains during the learning process to adapt automatically.

Our contributions, aiming to address these issues, are therefore two-fold:
- We extend the nearest class mean (NCM) classifier and its metric learning approach [14] to the problem of domain adaptation by introducing domain specific class means (DSCM) with domain specific weights. While this method yields surprisingly good results without any learning procedure (besides computing the per-class and per-domain means), we also show that metric learning yields further improvement. This involves learning a such transformation of the feature space that decreases interclass distances and increases intraclass distances independently of the domain (see illustration for two domains in Fig.1).
- Inspired by [19], we propose a self-adaptive metric learning domain adaptation (SaMLDa) framework where we exploit the available unlabeled target instances to better adjust the learned metric to the target domain. The idea is to use the DSCM classifier 1) to select and label unlabeled target instances to enrich the training set and 2) to remove the more ambiguous source examples from the training set. This dynamically updated training set is used to iteratively refine the learned transformation by forcing the learning process to exploit the characteristics of the unlabeled target instances. While naturally our SaMLDa framework uses the DSCM based metric learning approach, we show that other metric learning approaches can be used in the framework in order to improve the classification on the target instanced in the transformed space.

The rest of the paper is organized as follows. In section 2 we review the relevant literature. Then, in section 3 we describe the DSCM classifier and the corresponding metric learning and in section 4, the self-adaptive metric learning domain adaptation (SaMLDa) framework. Section 5 provides details about the datasets and our evaluation framework and section 6 details the results of our experiments. Finally we conclude the paper in section 7.

2 Related Work

The domain adaptation literature is very vast. We refer the interested reader to [12] for a survey of domain adaptation methods with a focus on learning theory and natural language processing applications, and to [2] for a survey focusing on computer vision applications. As for [15], it provides a review on the related topic of transfer learning.

We will limit our review on the most related works, *i.e.* those that focus on transforming the feature space in order to bring the domains closer. This class of methods can be further split into methods that do an unsupervised transformation, generally based on PCA projections, such as the subspace based domain adaptation methods in [1,6,7,9].

In contrast to these unsupervised learning of the transformation matrix, there are a set of metric learning based methods that exploits class labels (in general both in the source and in the target domain) to learn a transformation of the feature space such that in this new space the instances of the same class become closer to each other than to instances from other classes and this independently of the domain they belong to [10,13,17,21]. While the Domain Invariant Projections (DIP) in [1] are also learned without using class labels, the extended DIP+CC search for invariant projections that encourages samples with the same class labels to form a more compact cluster. The proposed DSCMML method can also be placed in this subgroup.

In addition we exploit unlabeled target instances and refine the metric accordingly. The proposed framework is inspired by [19], but the idea of using unlabeled target instance in an active supervised domain adaptation can be also found in [18]. Similarly, [5] proposes to use unlabeled target data, however they are used to measure the data distribution mismatch between the source and target domain. Their Domain Transfer SVM generalizes the sample re-weighting of Kernel Mean Matching [11] by simultaneously learning the SVM decision function and the kernel such that the difference between the means of the source and target feature distributions (including the unlabeled ones) are minimized.

3 Domain Specific Class Means Classifier

The Domain Specific Nearest Class Means (DSCM) classifier extends the NCM by considering domain based class means and it was inspired by the Nearest Class Multiple Centroids (NCMC) classifier proposed in [14]. In what follows, we first review the NCM and NCMC and then introduce the proposed DSCM.

The nearest class mean (NCM) classifier assigns an image to the class $c^* \in Y_c = \{1, \ldots, C\}$ whose mean is the closest:

$$c^* = \underset{c \in Y_c}{\operatorname{argmin}} \; d_{\boldsymbol{W}}(\boldsymbol{x}_i, \boldsymbol{\mu}_c), \quad \text{with} \quad \boldsymbol{\mu}_c = \frac{1}{C} \sum_{i:y_i=c} \boldsymbol{x}_i, \tag{1}$$

where y_i is the ground-truth label of the image $\boldsymbol{x}_i$, N^c is the number of training examples from the class c and $d_{\boldsymbol{W}}(\boldsymbol{x}_i, \boldsymbol{\mu}_c) = ||\boldsymbol{W}(\boldsymbol{x}_i - \boldsymbol{\mu}_c)||^2$ is the squared Euclidean distance between an instance $\boldsymbol{x}_i$ and the class mean $\boldsymbol{\mu}_c$ in some projected feature space given by the transformation matrix $\boldsymbol{W}$. If $\boldsymbol{W}$ is the identity ($\boldsymbol{I}$), this corresponds to the Euclidean distance in the original feature space.

We can easily reformulate Eq. 1 as a multi-class softmax assignment using a mixture model (with equal weights), where the probability that an image $\boldsymbol{x}$ belongs to the class c is defined as follows (see also [14]):

$$p(c|\boldsymbol{x}_i) = \frac{\exp\big(-\frac{1}{2}d_{\boldsymbol{W}}(\boldsymbol{x}_i, \boldsymbol{\mu}_c)\big)}{\sum_{c'=1}^{N_C} \exp\big(-\frac{1}{2}d_{\boldsymbol{W}}(\boldsymbol{x}_i, \boldsymbol{\mu}_{c'})\big)}. \tag{2}$$

and the final assignment is done by $c^* = \text{argmax}_{c \in Y_c}\, p(c|\boldsymbol{x}_i)$. This definition may also be interpreted as the posterior probabilities of a Gaussian generative model $p(\boldsymbol{x}_i|c) = \mathcal{N}(\boldsymbol{x}_i, \boldsymbol{\mu}^c, |\Sigma)$ with the mean $\boldsymbol{\mu}^c$ and class independent covariance matrix $\Sigma = (\boldsymbol{W}^\top \boldsymbol{W})^{-1}$.

Note that, once the metric $d_{\boldsymbol{W}}$ is known, learning the mean parameters of this classifier is very efficient as it only involves summing the image descriptors for each class and domain. A great advantage is that, if the metric is fixed, we can easily add new classes, new domains or new training images to existing classes and domains at almost zero cost [14].

The Nearest Class Multiple Centroids (NCMC) classifier [14] extends the NCM classifier by representing each class by a set of centroids $\{\boldsymbol{m}_c^j\}_{j=1}^k$ obtained by clustering images within each class, instead of a single class mean (NCM corresponds to $k = 1$ and $m_c^1 = \boldsymbol{\mu}_c$). The posterior probability for the class c is then defined as:

$$p(c|\boldsymbol{x}) = \frac{1}{Z}\sum_{j=1}^{k} \exp\big(-\frac{1}{2}d_W(\boldsymbol{x}, \boldsymbol{m}_c^j)\big), \tag{3}$$

where $Z = \sum_c \sum_j \exp\big(-\frac{1}{2}d_W(\boldsymbol{x}, \boldsymbol{m}_c^j)\big)$ is the normalizer and the model for each class becomes an equal weighted Gaussian mixture distribution with $\boldsymbol{m}_c^j$ as means and $\Sigma = (\boldsymbol{W}^\top \boldsymbol{W})^{-1}$ being the shared covariance matrix.

The Proposed Domain Specific Class Means (DSCM) Classifier. The NCMC classifier can be naturally extended to the case of multiple domains where instead of selecting centroids in an unsupervised manner, for each class we consider domain specific class means:

$$\boldsymbol{\mu}_d^c = \frac{1}{N_d^c} \sum_{i: y_i = c, x_i \in \mathcal{D}_d} \boldsymbol{x}_i, \tag{4}$$

where N_d^c is the number of images from class c in domain $\mathcal{D}_d$. In such a case, the class assignment can be written as:

$$p(c|\boldsymbol{x}_i) = \frac{\sum_d w_d \exp\big(-\frac{1}{2}d_{\boldsymbol{W}}(\boldsymbol{x}_i, \boldsymbol{\mu}_d^c)\big)}{\sum_{c'}\sum_d w_d \exp\big(-\frac{1}{2}d_{\boldsymbol{W}}(\boldsymbol{x}_i, \boldsymbol{\mu}_d^{c'})\big)} = \frac{\sum_d w_d p(\boldsymbol{x}_i|c, d)}{\sum_{c'}\sum_d w_d p(\boldsymbol{x}_i|c', d)} \tag{5}$$

This model also corresponds to a generative model, where the probability for an image $\boldsymbol{x}_i$ to be generated by class c is given by a Gaussian mixture distribution $p(\boldsymbol{x}_i|c) = \sum_d w_d p(x_i|c,d) = \sum_d w_d \mathcal{N}(\boldsymbol{x}_i, \mu_d^c, |\Sigma)$, with the mixing weights w_d, the domain specific class means μ_d^c and the class and domain independent covariance matrix $\Sigma = (\boldsymbol{W}^\top \boldsymbol{W})^{-1}$. Again, if $\boldsymbol{W} = \boldsymbol{I}$, the distances are computed in the original feature space, and $\Sigma = \boldsymbol{I}$.

Domain specific weights w_d in Eq. 5 allows to express different importance of different domains. These weights can be manually fixed (if we have some prior knowledge about the sources), learned (*e.g.* cross validated) or deduced directly from the data. If we denote the target domain by $\mathcal{T}$ and the source domains by $\mathcal{S}_i$, one possibility to define w_{s_i} is to measure how well the source domain $\mathcal{S}_i$ is aligned with the target domain $\mathcal{T}$ in the space projected by $\boldsymbol{W}$. This can be done by *e.g.* using target density around source ($TDAS$) proposed in [6]:

$$TDAS = \frac{1}{N_s} \sum_{\boldsymbol{x}_i^s \in \mathcal{S}} |\{\boldsymbol{x}^t | d_{\boldsymbol{W}}(\boldsymbol{x}^t, \boldsymbol{x}_i^s)| \leq \epsilon|, \tag{6}$$

where ϵ is set to the half of the mean distance between the source sample and the nearest target sample.

Alternatively, in the semi-supervised setting, where a small set of labeled instances are available from the target domain (denoted by $\mathcal{T}_l$), we can proceed as follows. We consider the class means for each source $\mathcal{S}_i$ individually and use them within Eq. 2 to predict the labels for instances in $\mathcal{T}_l$. The average classification accuracy of this classifier can be used directly as w_{s_i}.

Note that in the case of large datasets, Eq. 5 can be further extended by considering a set of domain specific prototypes for each class by clustering the N_d^c domain specific images from class c and domain d. Another possible extension is to consider domain and class specific weights w_d^c in Eq. 5, where both, the TDAS measure and the NCM accuracy can be also computed for each class individually. However these extensions being more suitable for larger datasets will not be considered in the paper.

3.1 Metric Learning for DSCM

The aim of metric learning is to find a projection matrix $\boldsymbol{W}$, such that the log-likelihood of the correct DSCM class predictions are maximized on X:

$$\mathcal{L} = \sum_{\boldsymbol{x}_i \in X} \ln p(c = y_i|\boldsymbol{x}_i) = \sum_{\boldsymbol{x}_i \in X} \Big[\ln \sum_d w_d p(\boldsymbol{x}_i|y_i, d) - \ln Z_i \Big]$$

where $Z_i = \sum_{c'} \sum_d w_d p(\boldsymbol{x}_i|c', d)$

Similarly to [14], we optimize this log-likelihood with mini-batch stochastic gradient descend (SGD) using a fixed learning rate (η), and randomly sampled small batch, $X_b \subset X$ of the training data to update $\boldsymbol{W}$ with the gradient:

$$\nabla_W \mathcal{L} = \sum_{\boldsymbol{x}_i \in X_b} \Big[\sum_{c'} \sum_d \Big(\frac{g_{i,d}^{c'}}{Z_i} - [\![c' = y_i]\!] \frac{g_{i,d}^{c'}}{p(\boldsymbol{x}_i|c')} \Big) \boldsymbol{W} (\boldsymbol{\mu}_d^{c'} - \boldsymbol{x}_i)(\boldsymbol{\mu}_d^{c'} - \boldsymbol{x}_i)^\top \Big]$$

where $g_{i,d}^{c'} = w_d p(x_i|c', d)$ and $[\![\cdot]\!]$ is one if its argument is true and zero otherwise. Note that we initialize $\boldsymbol{W}$ with principal component analyses, keeping the number of eigenvectors corresponding to the desired dimension of the projected space, generally, much smaller than the initial feature space.

4 Self-adaptive Metric Learning for Domain Adaptation

Two main cases are in general distinguished in domain adaptation. Unsupervised DA refers to the case where no labeled data is available from the target domain and semi-supervised DA where there are a few labeled images from the target domain to guide the learning process. Let denote by $\mathcal{T}_l$ the set of labeled target instances (that can be empty) and by $\mathcal{T}_u$ the set of unlabeled target set. Let $\mathcal{S}_1, \ldots, \mathcal{S}_{N_S}$ denote N_S source domains and X_r a current training set containing labeled instances (that can be ground truth labels or predicted ones) from different sources. We denote by Y_c the class labels, by $Y_d = \{s_1, \ldots s_{N_S}, t\}$ the domain-related labels, where t refers to the target domain and by $\boldsymbol{w}_d = (w_{s_1}, \ldots, w_t)$ the set of domain-specific weights.

We propose a Self-adaptive Metric Learning Domain Adaptation framework (SaMLDa) inspired by the method proposed in [19] where, similarly, we add iteratively unlabeled images from the target domain and remove images from the source to refine $\boldsymbol{W}$. The proposed framework assumes a metric learning component $f_W(X_r, \boldsymbol{W}_r, \boldsymbol{w}_d^r)$ that gets as input an initial transformation $\boldsymbol{W}_r$, a set of labeled training instances X_r and optionally a set of domain-specific weights $\boldsymbol{w}_d^r$. Then, using either only the class labels Y_c or also the domain-related labels Y_d of the instances in X_r, it outputs an updated transformation $\boldsymbol{W}_{r+1} = f_W(X_r, \boldsymbol{W}_r, \boldsymbol{w}_d^r)$.

The DSCMML described in section 3.1 is one particular case of the metric learning component, but the algorithm can use any other metric learning approach (and improve its performance as we will see in section 6). Indeed, in the case of metric learning methods not designed to handle multiple sources, the domain-related labels Y_d and weights $\boldsymbol{w}_d$ are simply ignored by f_W.

The main steps of our self-adaptive metric learning based domain adaptation algorithm (see Algorithm 1) are the followings:

- Using the initial labeled set X_1 (including labeled target instances if available), we set $\boldsymbol{W}_0$ as the first PCA directions on the training set and compute $\boldsymbol{W}_1 = f_{\boldsymbol{W}}(X_0, \boldsymbol{W}_0, \boldsymbol{w}_d^0)$. One advantage of the dimensionality reduction is that we have fewer parameters to learn, which is especially important when only a relatively small amount of training examples are available; it also generally leads to a better performance.
- Then, we iteratively refine $\boldsymbol{W}_r$ by adding at each iteration to the current training set X_r unlabeled instances from the target with their predicted class labels and removing the less confident source instances[1]. In contrast to [19],

[1] Improving in general the domain specific class means as the relevant source data becomes better clustered around these means.

Algorithm 1. The SaMLDa approach

Require: The initial training set $X_0 = \{\mathcal{S}_1, \ldots, \mathcal{S}_{N_S}, \mathcal{T}_l\}$.
Require: Domain-specific weights $\boldsymbol{w}_d^0$ and an initial transformation $\boldsymbol{W}_0$.
Ensure: A metric learning component $f_{\boldsymbol{W}}$.
1: Get $\boldsymbol{W}_1 = f_{\boldsymbol{W}}(X_1, \boldsymbol{W}_0, \boldsymbol{w}_d^0)$.
2: **for** $r = 1, \ldots, N_R$ **do**
3: Set $X_r = X_{r-1}$, $\boldsymbol{w}_d^r = w_d^{r-1}$ and compute μ_d^c.
4: Optionally, update $\boldsymbol{w}_d^r$ using $TDAS$ or NCM with $\boldsymbol{W}_r$.
5: For each $\boldsymbol{x}_i \in X_{r-1}$ and each class c_j compute $p(c_j|\boldsymbol{x}_i)$ using Eq. 5 with $\boldsymbol{W}_r$.
6: For each class c_j, add $\boldsymbol{x}_i^t \in \mathcal{T}_u$ to X_r for which $p(c^*|\boldsymbol{x}_i^t) - p(c^\dagger|\boldsymbol{x}_i^t)$ is the largest.
7: For each class c_j, remove $\boldsymbol{x}_j^s \in X_{r-1} \setminus \mathcal{T}$ from X_r for which $p(c^*|\boldsymbol{x}_j^s) - p(c^\dagger|\boldsymbol{x}_j^s)$ is the smallest.
8: Set $\boldsymbol{W}_{r+1} = f_{\boldsymbol{W}}(X_r, \boldsymbol{W}_r, \boldsymbol{w}_d^r)$.
9: If stopping criteria is met (classification accuracy degraded or no more data available to add or remove), quit the loop.
10: **end for**
11: Output $\boldsymbol{W}_{r^*}$ where $r^* + 1$ is the iteration at stopping criteria (or $r^* = N_R$).

where added or removed images are selected by the distances between low-level features[2], we use the DSCM class probabilities defined in Eq. 5.

1. For each class c_j, we add the unlabeled target example $\boldsymbol{x}_i^t \in \mathcal{T}_u$ to X_r for which $p(c^*|\boldsymbol{x}_i^t) - p(c^\dagger|\boldsymbol{x}^t{}_i)$ is the largest where $c^* = c_j$ is the predicted label of $\boldsymbol{x}^t{}_i$ and $p(c^\dagger|\boldsymbol{x}_i^t)$ is the second largest score. Note that as $\sum_{j=1}^{C} p(c_j|\boldsymbol{x}) = 1$, these are the images for which the classifier is the most confident about the prediction c^*, which however does not mean that the label is correct.
2. Similarly, for each class we remove the source image from X_r for which $p(c^*|\boldsymbol{x}_j^s) - p(c^\dagger|\boldsymbol{x}^s{}_j)$ is the smallest, *i.e.* the classifier finds it the most ambiguous example. Note that we use one element from all sources, but alternatively we could consider one per source decreasing more rapidly the amount of source data in X_r.
3. We compute $\boldsymbol{W}_{r+1} = f_{\boldsymbol{W}}(X_r, \boldsymbol{W}_r, \boldsymbol{w}_d^r)$ and optionally we update $\boldsymbol{w}_d^r$ as described in section 3.

– We iterate these steps until no more target data can be added, source data can be removed or the maximum iteration is achieved. However, adding training images with predicted labels comes with the risk of adding noise (incorrect labels). Therefore we also add the following stopping criteria. At each iteration, we evaluate the classification accuracy on the original labeled set and if the classification performance in step r incurs a stronger degradation than a predefined tolerance threshold (we used 1%) compared to the accuracy obtained in step r we stop iterating and retain $\boldsymbol{W}_r$, the metric obtained before degradation. Note that other criteria can also be considered such as measuring the variation between iterations of the $TDAS$.

[2] Their image-to-class distance is computed as a sum of distances between low level features extracted from the image and their closest low level feature within a class.

5 Datasets and Evaluation Framework

We used four datasets to test our framework: ICDA1 and ICDA2 from the ImageClef Domain Adaptation Challenge[3] as well as the OffCalSS and OffCalMS built on the Office dataset + Caltech10 used in several DA papers [1,6,8,9].

ICDA2. We denote by ICDA2 the dataset that was used in the challenge to submit the results. It consists of a set of image features[4] extracted by the organizers on randomly selected images collected from five different image collections: Caltech-256, ImageNet ILSVRC2012, PASCAL VOC2012, Bing and SUN denoted by C,I,P,B and S for simplicity. The organizers selected 12 common classes present in each datasets, namely, *aeroplane, bike, bird, boat, bottle, bus, car, dog, horse, monitor, motorbike, people*. Four collections (C, I, P and B) are proposed as source domains and for each of them 600 image feature and the corresponding labels were provided. The SUN dataset served as the target domain with 60 annotated and 600 non-annotated instances. The task was to provide predictions for the non-annotated target data. Neither the images nor the low-level features were accessible by the participants.

ICDA1. The ImageClef Domain Adaptation Challenge had two phases where in the first phase the participants were provided with a similar configuration as ICDA2, but with different features. We will denote this set with ICDA1. While in the case of ICDA2 we used only the provided train/test configuration and show the results obtained on the provided test set, in the case of ICDA1 we show the average results of an 11 fold cross-validation setting, where the 600 test documents were split into 10 folds and the train set added as 11th fold. At each time one of the 11 folds was added to the source sets to train the classifier and tested on the 600 remaining target documents.

OffCalSS. The OffCalSS dataset was built following the semi-supervised settings used in [6,8,9], *i.e.* 10 common classes (*backpack, touring-bike, calculator, head-phone, computer-keyboard, laptop-101, computer-monitor, computer-mouse, coffee-mug and video-projector*) were selected from four domains Amazon, Caltech-10, DSLR and Webcam (denoted by A,C,D and W). To build the training set 8 images from each class (when the source domain was D or W) and 20 images (when the source was A or C) was considered, to which 3 target instances per class were added. The training source and target examples were randomly selected and the experiment was repeated 20 times.

OffCalMS. In OffCalSS we evaluate only one domain (source) versus one domain (target). Also, the source domain considered is relatively small, which is, in some sense, a bit contradictory to the general assumption of domain adaptation, where it is in general assumed that large labeled source instances are

[3] http://www.imageclef.org/2014/adaptation
[4] SIFT based bag-of-visual words [3].

available to support the learning process. Therefore, using the same dataset and features as for OffCalSS, we build a multi-source setting similar to ICDA1 to which we will refer to as OffCalMS. In this case we fix one of the dataset as target (*e.g.* A) and use several or all the others (C,D and W) as source to evaluate our multi-source framework (AMLDA). In this case, all the labeled sources documents are used to which we add 3 randomly sampled target examples per class. We repeated this experiment 10 times and report average results.

In the case of multi-source datasets (ICDA1, ICDA2 and OffCalMS), we consider different source combinations by an exhaustive enumeration of all possible subsets. For instance, in the case of ICDA1, we have $SC_i, i = 1, \ldots, N_{SC}$, where $N_{SC} = 2^N_S - 1 = 15$ and N_S is the number of sources, *e.g.* in the case of ICDA1, $SC_1 = \{\mathbf{C}\}$, $SC_6 = \{\mathbf{C}, \mathbf{P}\}$ and $SC_{15} = \{\mathbf{C}, \mathbf{I}, \mathbf{P}, \mathbf{B}\}$. Then for each source combination, we concatenate the target training set $\mathcal{T}_l$ with the selected sources SC_j to build the training set X_1. If we denote the corresponding classifier by f^{SC_j}, to improve the final classification accuracy we can further combine the predictions of all these classifiers either using a majority vote or when available by averaging the class prediction scores. In both cases we used an unweighted combination in our experiments, but weighted fusion can also be used if there is enough data to learn the weights for each SC_i combination on a validation set. As we are in a semi-supervised setting we also consider f^{SC_0} in the combination, the classifier learned using only the labeled target set $\mathcal{T}_l$. We will denote the final prediction obtained as the combinations of all $N_{SC} + 1$ classifier by FusAll in the tables.

6 Experimental Results

Our first set of experiments were done on ICDA1 and ICDA2 in the semi-supervised setting. We consider all the available labeled source datasets (given a configuration) and the labeled target set grouped together as a single training set allowing us to consider different, not necessary multi-domain, classifiers to predict labels for the unlabeled target instances.

First, we consider the original feature space, meaning that we did not apply any metric learning procedure ($\boldsymbol{W} = \boldsymbol{I}$) and we evaluate classification performance of labeling the target examples of different source combinations.

Note that when we refer to the *original features*, they are different from the provided features as the latter were from the beginning power normalized[5] [16] using $\alpha = 0.5$. This allows already to an increase of 3-5% in accuracy on the baseline SVM and on the distance based classifiers, which explains why our baseline on OffCalSS is higher than in the literature.

As the SVM main baseline, for each configuration we considered the multi-class SVM from the LIBSVM package[6] trained in the original feature space. Using an initial 11 fold cross validation on ICDA1, we found that the linear kernel with $\nu = 0.12$, $C = 0.01$, and $\mu = 0.5$ performed the best. As only few

[5] As the original features are L1 normalized, we ended up with L2 normalized features.
[6] http://www.csie.ntu.edu.tw/~cjlin/libsvm/

Table 1. Classification performances on ICDA1 (left) and ICDA2 (right) in the original feature space, meaning that $\boldsymbol{W} = \boldsymbol{I}$ in the Eqs. 2, 3 and 5

ICDA1	SVM	KNN	NCM	NCMC	DSCM
C	26.32	22.79	25.08	23.83	**32.33**
I	31.85	25.92	23.71	21.71	**32.21**
P	27.32	18.91	23.83	21.06	**32.89**
B	**33.92**	27.98	27.83	27.23	33.36
C,I	29.08	22.7	23.48	21.21	**30.85**
C,P	27.29	20.09	23.03	21.36	**31.94**
C,B	30	26.52	26.94	25.2	**31.64**
I,P	29.88	19.33	24.42	20.68	**30.86**
I,B	**36.89**	27.85	24.91	23.14	30.94
P,B	30.12	22.48	26.83	23.02	**32.03**
C,I,P	28.67	20.05	24.65	20.52	**30.33**
C,I,B	**33.42**	25.79	24.44	21.55	30.17
C,P,B	28.05	22.44	26.26	22.74	**30.85**
I,P,B	**32.48**	22.91	25.59	22.39	30.59
C,I,P,B	29.39	22.38	24.89	22.06	**29.48**
Mean	30.31	23.21	25.06	23.21	**31.37**

ICDA2	SVM	KNN	NCM	NCMC	DSCM
C	23	22.5	11.67	13.17	**26**
I	**26.67**	25.5	13.00	16.33	25.5
P	**25.5**	20.67	19.50	15.17	24.83
B	**30.5**	24.17	15.33	14.83	24.67
C,I	22.67	23.50	13.5.0	13.17	**25.33**
C,P	21.83	21.67	16.00	11.33	**26.33**
C,B	26.00	23.00	12.50	13.83	**26.17**
I,P	**26.50**	17	14.33	20.33	**26.67**
I,B	**30**	23.33	15.67	13.83	27.17
P,B	**29.83**	22.17	15	22.17	26.5
C,I,P	22	22	14.17	12.5	**27.33**
C,I,B	**27.5**	21.5	14.67	18.67	24.33
C,P,B	24.17	22.67	15.83	18.67	**26.83**
I,P,B	**28.17**	21.33	15.5	15.83	27.67
C,I,P,B	24.5	21.5	16	17.5	**26.67**
Mean	25.92	22.44	14.84	22.44	**26.13**

target examples are in general available for each dataset, we used these fixed values for all datasets. Other parameters, such as the learning rate and number of iterations of the metric learning, were also cross-validated on ICDA1 and used on all datasets.

Evaluation in the Original Feature Space. First, we compare the SVM with distance based classifiers, namely KNN, NCM, NCMC and DSCM. As the first three are not domain adaptation specific methods they do not use the domain specific labels Y_d but consider the union of the labeled target and source instances as a single domain training set. In contrast, the Domain Specific Class Means (DSCM) classifier considers distances to class and domain specific class means, hence it is able to take advantage of domain specific labels Y_d. In this first set of experiments we use fixed weights (where $w_t = 2$ and $w_{s_i} = 1$). In the NCMC classifier, we use the same number of cluster means per class as the number of domains ($N_S + 1$) to fairly compare[7] with DSCM. The classification results are shown in Table 1. We can see that DSNCM outperforms all three distance based non parametric classifiers (KNN, NCM and NCMC) for all source combinations; it even outperforms for most configurations (and in average) the multi-class SVM. This already shows that DSCM applied without any learning is a suitable classifier for domain adaptation.

Evaluation in the Projected Feature Space. For experiments in the projected space, we consider metric learning approaches that optimizes $\boldsymbol{W}$ for the corresponding classifiers. For KNN we consider the metric learning where the ranking loss is optimized on triplets [4,20]:

$$L_{qpn} = \max(0, \left[1 + d_W(\boldsymbol{x}_q, \boldsymbol{x}_p) - d_W(\boldsymbol{x}_q, \boldsymbol{x}_n)\right]), \tag{7}$$

[7] The unsupervised clustering being does not guarantee the correspondence between clusters and domains. While different number of cluster centers might have yielded to better results, the training target dataset is rather small for cross-validation.

Table 2. Distance based classification performances on ICDA1 and ICDA2 using the features projected with $\boldsymbol{W}$ learned by the corresponding metric learning method

ICDA1	KNN +ML	NCM +ML	NCMC +ML	DSCM +ML
C	26.74	26.03	24.77	**28.41**
I	29.33	27.11	28.52	**32.88**
P	25.94	25.27	23.88	**26.5**
B	33.21	33.08	32.48	**34.85**
C,I	29.62	26.47	25.5	**30.89**
C,P	25.48	25.86	23.92	**30.32**
C,B	30.36	30.92	29.5	**32.92**
I,P	26.62	26	26.89	**31.86**
I,B	33.45	32.86	33.48	**35.23**
P,B	28.29	31.41	27.29	**33.68**
C,I,P	25.98	27.48	25.53	**31.67**
C,I,B	31.15	31.33	28.89	**34.68**
C,P,B	28.27	29.98	28.92	**32.67**
I,P,B	29.82	30.33	29.56	**34.58**
C,I,P,B	29.21	29.85	28.74	**33.24**
Mean	28.9	28.93	27.86	**32.29**

ICDA2	KNN +ML	NCM +ML	NCMC +ML	DSCM +ML
C	24.67	18.17	16.83	**25.17**
I	28.33	25.5	24.17	**30.33**
P	**26.33**	22.83	23.67	25.33
B	30.17	29	30.17	**34.17**
C,I	**25.83**	19.83	18	25.67
C,P	**24.83**	16.17	15	23.33
C,B	**27.83**	20.5	18.5	24.5
I,P	27.17	15	22.17	**29.67**
I,B	30.17	25.17	21.17	**33.5**
P,B	28.17	25.17	22.5	**30.67**
C,I,P	25	15.5	16.67	**27.17**
C,I,B	28.33	18.83	23.83	**28.5**
C,P,B	25.17	19.5	27.5	**27.83**
I,P,B	29.67	22	27.33	**31.83**
C,I,P,B	**27.83**	21.5	24.83	27.67
Mean	27.3	20.98	22.16	**28.36**

where $\boldsymbol{x}_p$ is an image from the same class as the query image $\boldsymbol{x}_q$, and $\boldsymbol{x}_n$ an image from any other class. The Nearest Class Mean Metric Learning (NCM+ML) optimizes $\boldsymbol{W}$ according to Eq. 2 and the Nearest Class Multiple Centroids classifier based metric learning (NCMC+ML) according to Eq. 3 (see details in [14]). Finally, we consider the Domain Specific Class Means based Metric Learning (DSCM+ML) described in section 3.1. We report results in Table 2. SVM results in the projected space are not included as we observed in general a drop[8]. in performance compared to Table 1. From these results, we can see that:

- Metric learning significantly improves the classification in the target domain in all cases, even when we apply methods which are not domain specific as ML for KNN, NCM and NCMC. The reason is that on the merged dataset the learning approach is able to take advantage of the class labels to bring closer the images from the same class independently of the domains and hence the final classifier is able to better exploit labeled data from the sources in the transformed space than in the original one.
- When we compare the different metric learning approaches, DSCMML outperforms all other methods on ICDA1 and in most cases on ICDA2. The few exceptions are when KNN+ML performs better than DSCM+ML on ICDA2. Note however that for ICDA2 we have a single test set, while on ICDA1 we average on 11 folds hence we can assume that DSCM+ML is consistently better than KNN+ML. Comparing to the SVM baseline (see Table 1) we see that DSCM+ML is almost always significantly better than the results obtained with the linear multi-class SVM.

Evaluation of SaMLDa with Different Metric Learning Algorithms. The aim of these experiments is to show that the Self-adaptive Metric Learning

[8] Reducing interclass and increasing intraclass distances does not mean improving linear separability between classes, especially when we decrease the dimensionality. Testing different non-linear kernels was out of the scope of the paper.

Table 3. Showing the improved accuracy for each metric learning when we refine the metric with the SaMLDa algorithm

ICDA1	KNN + ML	KNN + SaMLDa	NCM + ML	NCM + SaMLDa	NCMC + ML	NCMC + SaMLDa	DSCM+ ML	DSCM+ SaMLDa
C	26.74	**27.41**	26.03	**27.45**	24.77	**25.7**	28.41	**28.67**
I	29.33	**29.67**	27.11	**27.89**	28.52	**28.56**	32.88	**32.68**
P	25.94	**26.59**	25.27	**26.41**	23.88	**24.79**	26.5	**27.92**
B	33.21	**33.83**	33.08	**34.35**	32.48	**33.12**	34.85	**35.55**
C,I	29.62	**30.09**	26.47	**28.42**	25.5	**25.98**	30.89	**31.21**
C,P	25.48	**26.42**	25.86	**27.79**	23.92	**24.86**	30.32	**32**
C,B	30.36	**31.03**	30.92	**32.97**	29.5	**29.95**	32.92	**34.59**
I,P	26.62	**27.77**	26	**26.92**	**26.89**	26.65	31.86	**32.33**
I,B	33.45	**34.02**	32.86	**35.27**	33.48	**34.02**	35.23	**37.42**
P,B	28.29	**28.58**	31.41	**33.32**	27.29	**27.8**	33.68	**35.3**
C,I,P	25.98	**27.15**	27.48	**29.27**	25.53	**26**	31.67	**33.77**
C,I,B	31.15	**31.77**	31.33	**32.91**	28.89	**29.74**	34.68	**36.52**
C,P,B	**28.27**	28.21	29.98	**32.03**	28.92	**29.15**	32.67	**34.8**
I,P,B	29.82	**30.88**	30.33	**33.18**	29.56	**31.03**	34.58	**36.52**
C,I,P,B	29.21	**30.06**	29.85	**32.12**	28.74	**29.64**	33.24	**35.74**
Mean	28.9	**29.57**	28.93	**30.69**	27.86	**28.47**	32.29	**33.67**

Domain Adaptation (SaMLDa) described in section 4 can be used to further improve the performance of any of the previously mentioned metric learning approaches by iteratively updating the metric with unlabeled target examples. Note that in our algorithm the metric yielding the results in Table 2 correspond to the results obtained with $\boldsymbol{W}_1$. We also tested the performance with $\boldsymbol{W}_0$ corresponding to the PCA projection, but the results were far below the results obtained with $\boldsymbol{W}_1$. In Table 3 we compare the classification accuracies between a given metric learning $\boldsymbol{W}_1$ using only the initial training set X_1 and the metric $\boldsymbol{W}_{r^*}$ refined with SaMLDa, where $f_{\boldsymbol{W}}$ is the corresponding metric learning. We only show results on ICDA1, but similar behavior was observed on ICDA2.

The results show that if we integrate any of these metric learning approaches with the proposed SaMLDa algorithm, we are able to improve the classification accuracy in 58 out of 60 cases and for the two remaining cases the drop is not significant. When we compare SaMLDa with different metrics, best results are obtained when ML for DSCM is used.

Comparing Different Weighting Strategies. In Table 4 we compare different weighting strategies: fixed weights, weights obtained using $TDAS$ and weights using NCM accuracies. In the top 3 rows we show results when the weighting strategy was used in SaMLDa, *i.e.* we updated $\boldsymbol{w}_d^r$ at each iteration, while on the bottom 3 lines we show results when we used the manually fixed weights during the learning and used the $TDAS$ or NCM based weights with the learned metric $\boldsymbol{W}$ only at test time. In all cases we show the mean of all configuration results (as in the tables above), the results for all four sources and the results obtained as a late fusion of all SC_i source combinations (including SC_0). From Table 4 we can draw the following conclusions:

- The best weighting strategy is in general using the NCM accuracies. Using TDAS actually decreases the performance in most cases.
- Using the weighting strategy only at test time and using fix weight at training time seems to be a good compromise as the results are relatively similar and

Table 4. Different weighting strategies on ICDA1 (left) and ICDA2 (right) used during both training and test (top 3 rows) and test only (bottom 3 rows)

ICDA1	fix	TDAS	NCM
Mean	32.29	31.75	**32.67**
C,I,P,B	33.24	31.83	**34.53**
FusAll	39.18	39.29	**39.86**
Mean	32.29	31.69	**32.88**
C,I,P,B	33.24	32.73	**34.56**
FusAll	39.18	38.74	**39.56**

ICDA2	fix	TDAS	NCM
Mean	27.06	**28.39**	27.17
C,I,P,B	**30.67**	29	27.67
FusAll	**38**	37.17	37.5
Mean	**27.06**	26.72	27
C,I,P,B	30.67	28.83	**33**
FusAll	**38**	37.83	37.67

Table 5. Results on OffCalSS compared with [6] and [1], where for $\mathcal{D}_1 \rightarrow \mathcal{D}_2$, the domain $\mathcal{D}_1$ was used as source and $\mathcal{D}_2$ was the target

	SVM (ours)	DSCM	DSCM + SaMLDa	SA [6] +SVM	DIP+CC [1] +SVM	KNN (ours)	KNN + SaMLDa	SA [6] + KNN
C→A	53.66	50.64	54.14	44.7	**61.8**	42.01	53.02	45.3
D→A	46.37	48.76	46.77	41.6	**56.9**	43.08	47.34	45.8
W→A	44.72	48.43	45.66	39.3	**53.4**	40.33	46.52	44.8
A→C	44.6	34.89	44.21	40.6	**47.8**	33.75	43.96	38.4
D→C	38.5	34.24	38.62	34.8	**44.2**	31.58	38.73	35.8
W→C	36.79	33.42	36.59	32.6	**43.6**	29.19	36.8	34.1
A→D	50.94	62.05	53.07	40.9	**67.5**	50.87	54.09	55.1
C→D	54.57	61.57	56.97	41.1	**65.8**	50.71	55.51	56.6
W→D	76.81	64.65	73.27	77.6	**92.6**	71.65	77.2	82.3
A→W	53.68	66.08	56.81	38.2	**72.5**	56.77	59	60.3
C→W	54.42	65.06	59.58	**82.2**	69.9	58.17	58.74	60.7
D→W	83.74	71.47	81	87.1	**89.1**	78.79	83.53	84.8

in this way we can keep the training costs lower (no need to estimate the weights at each step).

– Note finally, that averaging the predictions from all source combinations (FusAll) improves the final results significantly in all cases compared to using the C, I, P, B source combination alone.

Evaluation on OffCalSS. We consider the OffCalSS dataset where in contrast to the other datasets we have only a single source and only a relatively small amount of training source images (20 or 8 instances from each class) to which 3 labeled target images per class were added from the target domain. For each source-target pair, we consider in the original feature space the multi-class SVM, the DSCM (with the NCM accuracy based weighting) and the KNN classifiers (with $k = 1$ as 1NN is used in general with this dataset in the literature). In addition, both for KNN and DSCM we show in Table 5 the results of the SaMLDa framework with the corresponding metric learning. As a comparison we show that our results are slightly better than the results of the unsupervised subspace alignment (SA) in [6], but they are below the current state-of-the art results on this dataset (DIP+CC) [1]. The latter also exploits the idea of bringing closer instances from the same class in the projected (latent) space, but exploits jointly with the empirical distance between the source and target domains when they learn the latent space.

Evaluation on OffCalMS. We finally evaluate the proposed methods on the OffCalMS where each dataset serve as target in turn and the others as sources

Table 6. Results on OffcalMS considering different dataset as target domains

A	SVM	DSCM	DSCM + SaMLDa	KNN	KNN + SaMLDa
A	47.53	47.07	**48.38**	43.32	43.32
C	56.76	50.78	**59.19**	43.18	54.62
D	44.11	**50.54**	46.24	42.46	47.35
W	41.63	**47.39**	42.13	39.83	44.66
C,D,W	57.22	52.59	**59.44**	43.5	53.79
FusAll	73.27	**80.57**	79.46	60.84	71.27

C	SVM	DSCM	DSCM + SaMLDa	KNN	KNN + SaMLDa
C	**33.39**	32.66	**33.39**	28.04	28.04
A	**46.76**	35.21	46.74	35.66	45.57
D	37.93	34.19	38.53	30.67	**39.79**
W	38.08	36	36.77	30.7	**39.21**
A,D,W	**50.84**	38.59	49.86	36.8	47.22
FusAll	66.4	61.48	**73.9**	46.44	64.85

D	SVM	DSCM	DSCM + SaMLDa	KNN	KNN + SaMLDa
D	56.61	58.82	**59.45**	56.06	56.06
A	46.38	**60.31**	50.71	52.52	47.87
C	47.48	**61.5**	50.87	51.26	49.84
W	87.64	68.19	85.04	**88.82**	**88.82**
A,C,W	**70.71**	70.47	70.16	77.95	65.04
FusAll	89.69	84.49	**95.28**	91.02	87.95

W	SVM	DSCM	DSCM + SaMLDa	KNN	KNN + SaMLDa
W	63.92	62.98	**64.6**	61.62	61.62
A	51.21	**67.58**	51.66	57.58	49.47
C	47.7	**66.57**	56.3	53.02	48.08
D	**86.24**	73.89	84.04	87.36	88.15
A,C,D	62.15	72.15	67.51	**78.68**	59.7
FusAll	90.64	92.34	95.36	**96**	89.74

where the whole dataset is considered for training. In Table 6 we show the results for 1) using only the labeled target set as training SC_0, 2) using each source individually ($SC_i, i = 1..3$) as training where $\mathcal{T}_l$ was added to the source, 3) using the set of all sources (SC_{N_S}), and finally 4) the fusion of all combination results (FusAll). In all cases we show similarly to OffCalSS the SVM, DSCM and KNN results in the original space and the results of the SaMLDa framework with the corresponding metric learning. Note that the test sets are the same as in Table 5 (up to the randomly selected small $\mathcal{T}_l$ set) so we can observe an important gain in performance due to adding more source instances and especially considering more than a single source domain (FuseAll).

7 Conclusion

Targeting multi-source domain adaptation we extended the Nearest Class Mean (NCM) classifier by introducing, for each class, domain-dependent mean parameters as well as domain-specific weights. We have shown that the proposed Domain Specific Class Means (DSCM) classifier is already suitable for domain adaptation without any learning and its performance can be further improved by appropriately designed metric learning. As a second contribution, which is orthogonal to the first one and therefore complementary, was a generic self-adaptive metric learning technique that iteratively curates the training set by adding unlabeled samples for which the prediction confidence was high and removing the labeled samples for which the prediction confidence was low. We have shown on two public benchmarks, the ImageClef Domain Adaptation Challenge and the Office-CalTech datasets, that the proposed self-adaptive metric learning approach can bring improvement to various metric learning approaches in the domain adaptation framework.

References

1. Baktashmotlagh, M., Harandi, M.T., Lovell, B.C., Salzmann, M.: Unsupervised domain adaptation by domain invariant projection. In: ICCV (2013)

2. Beijbom, O.: Domain adaptations for computer vision applications. University of California, San Diego (June (2012)
3. Csurka, G., Dance, C., Fan, L., Willamowski, J., Bray, C.: Visual categorization with bags of keypoints. In: SLCV (ECCV Workshop) (2004)
4. Davis, J.V., Kulis, B., Jain, P., Sra, S., Dhillon, I.S.: Information-theoretic metric learning. In: ICML (2007)
5. Duan, L., Tsang, I.W., Xu, D., Maybank, S.J.: Domain transfer SVM for video concept detection. In: CVPR (2009)
6. Fernando, B., Habrard, A., Sebban, M., Tuytelaars, T.: Unsupervised visual domain adaptation using subspace alignment. In: ICCV (2013)
7. Gong, B., Shi, Y., Sha, F., Grauman, K.: Geodesic flow kernel for unsupervised domain adaptation. In: CVPR (2012)
8. Gong, B., Grauman, K., Sha, F.: Reshaping visual datasets for domain adaptation. In: NIPS (2013)
9. Gopalan, R., Li, R., Chellappa, R.: Domain adaptation for object recognition: An unsupervised approach. In: ICCV (2011)
10. Hoffman, J., Kulis, B., Darrell, T., Saenko, K.: Discovering Latent Domains for Multisource Domain Adaptation. In: Fitzgibbon, A., Lazebnik, S., Perona, P., Sato, Y., Schmid, C. (eds.) ECCV 2012, Part II. LNCS, vol. 7573, pp. 702–715. Springer, Heidelberg (2012)
11. Huang, J., Smola, A., A., Borgwardt, K., Schoelkopf, B.: Correcting sample selection bias by unlabeled data. In: NIPS (2007)
12. Jiang, J.: A literature survey on domain adaptation of statistical classifiers. Tech. rep. (2008)
13. Kulis, B., Saenko, K., Darrell, T.: What you saw is not what you get: Domain adaptation using asymmetric kernel transforms. In: CVPR (2011)
14. Mensink, T., Verbeek, J., Perronnin, F., Csurka, G.: Distance-based image classification: Generalizing to new classes at near zero cost. PAMI **35**(11), 2624–2637 (2013)
15. Pan, S.J., Yang, Q.: A survey on transfer learning. IEEE Transactions on Knowledge and Data Engineering **22**(10), 1345–1359 (2010)
16. Perronnin, F., Sánchez, J., Mensink, T.: Improving the Fisher Kernel for Large-Scale Image Classification. In: Daniilidis, K., Maragos, P., Paragios, N. (eds.) ECCV 2010, Part IV. LNCS, vol. 6314, pp. 143–156. Springer, Heidelberg (2010)
17. Saenko, K., Kulis, B., Fritz, M., Darrell, T.: Adapting Visual Category Models to New Domains. In: Daniilidis, K., Maragos, P., Paragios, N. (eds.) ECCV 2010, Part IV. LNCS, vol. 6314, pp. 213–226. Springer, Heidelberg (2010)
18. Saha, A., Rai, P., Daumé III, H., Venkatasubramanian, S., DuVall, S.L.: Active Supervised Domain Adaptation. In: Gunopulos, D., Hofmann, T., Malerba, D., Vazirgiannis, M. (eds.) ECML PKDD 2011, Part III. LNCS, vol. 6913, pp. 97–112. Springer, Heidelberg (2011)
19. Tommasi, T., Caputo, B.: Frustratingly easy nbnn domain adaptation. In: ICCV (2013)
20. Weinberger, K., Saul, L.: Distance metric learning for large margin nearest neighbor classification. JMLR **10**, 207–244 (2009)
21. Zha, Z.J., Mei, T., Wang, M., Wang, Z., Hua, X.S.: Robust distance metric learning with auxiliary knowledge. In: IJCAI (2009)