From N to N+1: Multiclass Transfer Incremental Learning

Ilja Kuzborskij[1,2], Francesco Orabona[3], and Barbara Caputo[1]

[1] Idiap Research Institute, Switzerland,
[2] École Polytechnique Fédérale de Lausanne (EPFL), Switzerland,
[3] Toyota Technological Institute at Chicago, USA,

ilja.kuzborskij@idiap.ch, francesco@orabona.com, barbara.caputo@idiap.ch

Abstract

Since the seminal work of Thrun [17], the learning to learn paradigm has been defined as the ability of an agent to improve its performance at each task with experience, with the number of tasks. Within the object categorization domain, the visual learning community has actively declined this paradigm in the transfer learning setting. Almost all proposed methods focus on category detection problems, addressing how to learn a new target class from few samples by leveraging over the known source. But if one thinks of learning over multiple tasks, there is a need for multiclass transfer learning algorithms able to exploit previous source knowledge when learning a new class, while at the same time optimizing their overall performance. This is an open challenge for existing transfer learning algorithms. The contribution of this paper is a discriminative method that addresses this issue, based on a Least-Squares Support Vector Machine formulation. Our approach is designed to balance between transferring to the new class and preserving what has already been learned on the source models. Extensive experiments on subsets of publicly available datasets prove the effectiveness of our approach.

1. Introduction

Vision-based applications like Google Goggle, assisted ambient living, home robotics and intelligent car driver assistants all share the need to distinguish between several object categories. They also share the need to update their knowledge over time, by learning new category models whenever faced with unknown objects. Consider for instance the case of a service robot, designed for cleaning up kitchens in public hospitals. Its manufacturers will have equipped it with visual models of objects expected to be found in a kitchen, but inevitably the robot will encounter something not anticipated at design time – perhaps an object out of context, such as a personal item forgotten by a patient on her food tray, or a new type of food processor that entered the market after the robot. To learn such new object, the robot will generally have to rely on little data and explanation from its human supervisor. Also, it will have to preserve its current range of competences while adding the new object to its set of known visual models. This challenge, which holds for any intelligent system equipped with a camera, can be summarized as follows: suppose you have a system that knows N objects (source). Now you need to extend its object knowledge to the $N+1$-th (target), using only few new annotated samples, without having the possibility to re-train everything from scratch. Can you add effectively the new target $N+1$-th class model to the known N source models by leveraging over them, while at the same time preserving their classification abilities?

As of today, we are not aware of previous work addressing this issue, nor of existing algorithms able to capture all its nuances. The problem of how to learn a new object category from few annotated samples by exploiting prior knowledge has been extensively studied [20, 11, 7]. The majority of previous work focused on object category detection (i.e. binary classification) rather than the multiclass case [1, 19, 18]. It is natural to ask if such previous methods would work well in the scenario depicted, by just extending them to the multiclass. We argue that to solve the $N \longrightarrow N+1$ transfer learning problem one needs to address a deeper algorithmic challenge.

In addition, learning from scratch and preserving training sets from all the source tasks might be infeasible due to the large number of tasks or when acquiring tasks incrementally, especially for large datasets [15]. In object categorization case this might come as training source classifiers from large scale visual datasets, in abundance of data.

Consider the following example: a transfer learning task of learning a dog detector, given that the system has already

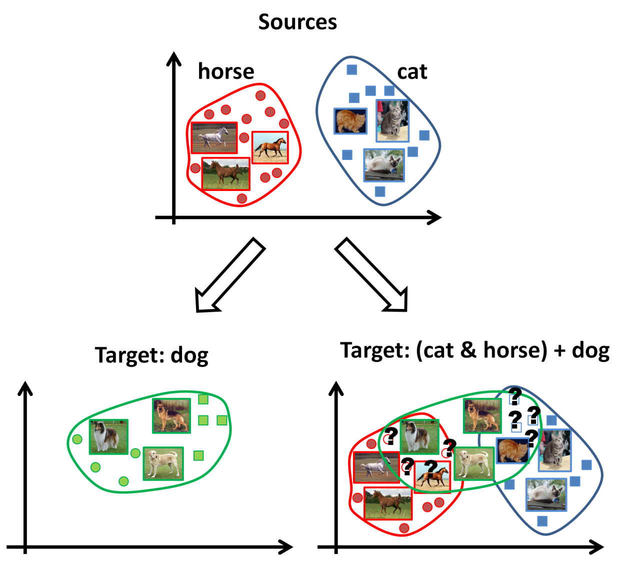

Figure 1: Binary (left) versus $N \longrightarrow N+1$ transfer learning (right). In both cases, transfer learning implies that the target class is learned close to where informative sources models are. This is likely to affect negatively performance in the $N \longrightarrow N+1$ case, where one aims for optimal accuracy on the sources and target classes simultaneously.

learned other kind of animal detectors. This is achieved, in one form or another, by constraining the dog model to be somehow "similar" to the horse and cat detectors learned before [11, 18]. Success in this setting is defined as optimizing the accuracy of the dog detector, with a minimal number of annotated training samples (Figure 1, left).

But if we consider the multiclass case, the different tasks now "overlap". Hence we are faced with two opposite needs: on one side, we want to learn to recognize dogs from few samples, and for that we need to impose that the dog model is close to the horse and cat models learned before. On the other side, we want to optimize the overall system performance, which means that we need to avoid mispredictions between classes at hand (Figure 1, right). These two seemingly contradictory requirements are true for many $N \longrightarrow N+1$ transfer learning scenarios: how to reconcile them in a principled manner is the contribution of this paper.

We build on the algorithm of Tommasi et al. [18], a transfer learning method based on the multiclass extension of Least-Squares Support Vector Machine (LSSVM) [16]. Thanks to the linear nature of LSSVM, we cast transfer learning as a constraint for the classifier of the $N+1$ target class to be close to a subset of the N source classifiers. At the same time, we impose a stability to the system, biasing the formulation towards solutions close to the hyperplanes of the N source classes. In practice, given N source models, we require that these models would not change much when going from N to $N+1$.

As in [18], we learn how much to transfer from each of the source classifiers, by minimizing the Leave-One-Out (LOO) error, which is an unbiased estimator of the generalization error for a classifier [4]. We call our algorithm MULticlass Transfer Incremental LEarning (MULTIpLE).

Experiments on various subsets of the Caltech-256 [9] and Animals with Attributes (AwA) datasets [13] show that our algorithm outperforms the One-Versus-All (OVA) extension of [18], as well as other baselines [11, 20, 1]. Moreover, its performance often is comparable to what it would be obtained by re-training the whole $N+1$ classifier from all data, without the need to store the source training data.

The paper is organized as follows: after a review of previous work (Section 2), we describe our setting (Section 3) and our algorithm (Section 4). Experiments are reported in Section 5, followed by conclusions in Section 6.

2. Related Work

Prior work in transfer learning addresses mostly the binary classification problem (object detection). Some approaches transfer information through samples belonging to both source and target domains during the training process, as in [14] for reinforcement learning. Feature space approaches consider transferring or sharing feature space representations between source and target domains. Typically, in this setting source and target domain samples are available to the learner. In that context, Blitzer *et al.* [2] proposed a heuristic for finding corresponding features, that appear frequently in both domains. Daumé [6] showed a simple and effective way to replicate feature spaces for performing adaptation for the case of natural language processing. Yao and Doretto [20] proposed an AdaBoost-based method using multiple source domains for the object detection task.

Another research line favors model-transfer (or parameter-transfer) methods, where the only knowledge available to the learner is "condensed" within a model trained on the source domain. Thus, samples from source domain are not preserved. Model-transfer is theoretically sound as was shown by Kuzborskij and Orabona [12], since relatedness of the source and target tasks enables quick convergence of the empirical error estimate to the true error. Within this context, Yang *et al.* [19] proposed a kernelizable SVM-like classifier with a biased regularization term. There, instead of the standard ℓ_2 regularization, the goal of the algorithm is to keep the target domain classifier "close" to the one trained on the source domain. Tommasi *et al.* [18] proposed a multi-source transfer model with a similar regularizer, where each source classifier was weighted by learned coefficients. The method obtained strong results on the visual object detection task, using only a small amount of samples from the target domain. Aytar and Zisserman [1] proposed a similar model, with a linear formulation for the problem of object localization. Both methods rely on weighted source classifiers, which is crucial when attempting to avoid negative transfer. Several Multiple Kernel Learning (MKL) methods were proposed for solving transfer learning problems. Jie *et al.* [11] suggested to use MKL kernel weights as source classifier weights, proposing one of the few truly multiclass transfer learning models. An MKL approach was also proposed by Duan *et al.* [7]. There, kernel weights affect both the source classifiers and the representation of the target domain.

3. Problem Setting and Definitions

In the following we denote with small and capital bold letters respectively column vectors and matrices, e.g. $\boldsymbol{\alpha} = [\alpha_1, \alpha_2, \ldots, \alpha_N]^T \in \mathbb{R}^N$ and $\boldsymbol{A} \in \mathbb{R}^{M \times N}$ with A_{ji} corresponding to the (j,i) element. When only one subscripted index is present, it represents the column index: e.g. $\boldsymbol{A}_i$ is the i-th column of the matrix $\boldsymbol{A}$.

As in related literature, we define a set of M training samples consisting of a feature vector $\boldsymbol{x}_i \in \mathbb{R}^d$ and the corresponding label $y_i \in \mathcal{Y} = \{1, \ldots, N, N+1\}$ for $i \in \{1, 2, \ldots, M\}$. We will denote by $\boldsymbol{X}$ the sample-column matrix, i.e. $\boldsymbol{X} = [\boldsymbol{x}_1, \cdots, \boldsymbol{x}_M]$. We use the formalism of linear classifiers, so that a multiclass classifier is described as a matrix $\boldsymbol{W} = [\boldsymbol{W}_1, \ldots, \boldsymbol{W}_N]$, where each column vector $\boldsymbol{W}_n$ represents the hyperplane that separates one of the N classes from the rest. Hence, the label associated to a given sample $\boldsymbol{x}$ is predicted as $f_{\boldsymbol{W}}(\boldsymbol{x}) := \underset{n=1,\ldots,N}{\operatorname{argmax}}\ \boldsymbol{W}_n^\top \boldsymbol{x} + b_n$. Note that it is straightforward to lift the theory to the non-linear domain by using kernels, and for clarity we describe the algorithm in linear notation.

A common way to find the set of hyperplanes $\boldsymbol{W}$ is by solving a regularized problem with a convex loss function, that upper bounds the 0/1 loss. For reasons that will become clear later, we base our method on the OVA variant of the LSSVM [16], which combines a square loss function with ℓ_2 regularization. Defining the label matrix $\boldsymbol{Y}$ such that Y_{in} is equal to 1 if $y_i = n$ and -1 otherwise, we obtain the multiclass LSSVM objective function

$$\min_{\boldsymbol{W}, \mathbf{b}} \frac{1}{2} \|\boldsymbol{W}\|_F^2 + \frac{C}{2} \sum_{i=1}^{M} \sum_{n=1}^{N} (\boldsymbol{W}_n^\top \boldsymbol{x}_i + b_n - Y_{in})^2,$$

where $\|\cdot\|_F$ is the Frobenius norm.

In our setting of interest, there are two types of information. First, we have a set of models that were obtained from the source N class problem. These source models are encoded as a set of N hyperplanes, that we again represent in matrix form as $\boldsymbol{W}' = [\boldsymbol{W}'_1, \ldots, \boldsymbol{W}'_N]$. Note that we assume no access to the samples used to train the source classifiers. Second, we have a small training set composed from samples belonging to all the $N+1$ classes, target and source classes.

4. MULTIpLE

The aim of our approach is to find a new set of hyperplanes $\boldsymbol{W} = [\boldsymbol{W}_1, \ldots, \boldsymbol{W}_N], \boldsymbol{w}_{N+1}$, such that i) performance on the target $N+1$-th class improves by transferring from the source models, and ii) performance on the source N classes should not deteriorate or even improve compared to the former. Thanks to the model linearity, we obtain a metric between classifiers, that could be used to find classifiers with similar performance by enforcing the distance between them to be small. We propose to achieve both aims above through the use of distance-based regularizers.

The first objective can be recognized as the transfer learning problem. It has been shown that this can be implemented using the regularizer $\|\boldsymbol{w}_{N+1} - \boldsymbol{W}'\boldsymbol{\beta}\|^2$ [18]. This term enforces the target model $\boldsymbol{w}_{N+1}$ to be close to a linear combination of the source models, while negative transfer is prevented by weighing the amount of transfer of each source model using the coefficient vector $\boldsymbol{\beta} = [\beta_1 \ldots \beta_N]^\top$. The second objective of avoiding degradation of existing models $\boldsymbol{W}'$ has been ignored in the transfer learning literature. However, as explained before, adding a target class may affect the performance of the source models and it is therefore useful to transfer the novel information back to the N source models. To prevent negative transfer, we enforce the new hyperplanes $\boldsymbol{W}$ to remain close to the source hyperplanes $\boldsymbol{W}'$ using the term $\|\boldsymbol{W} - \boldsymbol{W}'\|_F^2$. With both regularizers in the LSSVM objective function, we obtain

$$\begin{aligned}\min_{\boldsymbol{W}, \boldsymbol{w}_{N+1}, \mathbf{b}} \quad & \frac{1}{2}\|\boldsymbol{W} - \boldsymbol{W}'\|_F^2 + \frac{1}{2}\|\boldsymbol{w}_{N+1} - \boldsymbol{W}'\boldsymbol{\beta}\|_F^2 \\ & + \frac{C}{2} \sum_{i=1}^{M} \sum_{n=1}^{N+1} (\boldsymbol{W}_n^\top \boldsymbol{x}_i + b_n - Y_{in})^2 .\end{aligned}$$

The solution to this minimization problem is given by

$$\boldsymbol{W}_n = \boldsymbol{W}'_n + \sum_{i=1}^{M} A_{in} \boldsymbol{x}_i, \ n = 1, \cdots, N \tag{1}$$

$$\boldsymbol{w}_{N+1} = \sum_{n=1}^{N} \beta_n \boldsymbol{W}'_n + \sum_{i=1}^{M} A_{i(N+1)} \boldsymbol{x}_i, \tag{2}$$

and $\mathbf{b} = \mathbf{b}' - \left[\mathbf{b}'' \ \ \mathbf{b}''^\top \boldsymbol{\beta}\right]$, where $\boldsymbol{A} = \boldsymbol{A}' - [\boldsymbol{A}'' \ \ \boldsymbol{A}''\boldsymbol{\beta}]$,

$$\begin{bmatrix} \boldsymbol{A}' \\ \mathbf{b}'^\top \end{bmatrix} := \mathbf{M} \begin{bmatrix} \mathbf{Y} \\ \mathbf{0} \end{bmatrix} \tag{3}$$

$$\begin{bmatrix} \boldsymbol{A}'' \\ \mathbf{b}''^\top \end{bmatrix} := \mathbf{M} \begin{bmatrix} \mathbf{X}^\top \mathbf{W}' \\ \mathbf{0} \end{bmatrix} \tag{4}$$

$$\mathbf{M} := \begin{bmatrix} \mathbf{X}^\top \mathbf{X} + \frac{1}{C}\mathbf{I} & \mathbf{1} \\ \mathbf{1}^\top & 0 \end{bmatrix}^{-1} . \tag{5}$$

The solution of the tranfer learning problem is completely defined once we set the parameters $\boldsymbol{\beta}$. In the next section we describe how to automatically tune these parameters.

4.1. Self-tuning of Transfer Parameters

We want to set the transfer coefficients $\boldsymbol{\beta}$ to improve the performance by exploiting only relevant source models while preventing negative transfer. With this in mind, we extend the method of [18] to our setting and to our objective function. We optimize the coefficients $\boldsymbol{\beta}$ automatically using an objective based on the LOO error, which is an almost unbiased estimator of the generalization error of a classifier [4]. An advantage of LSSVM over other methods is that it allows the LOO error to be computed efficiently in closed-form.

Specifically, we cast the optimization of $\boldsymbol{\beta}$ as the minimization of a convex upper bound of the LOO error. The LOO prediction for a sample i with respect to hyperplane $\boldsymbol{W}_n$ is given by (derivation is available in supplementary material[1])

$$\tilde{Y}_{in}(\boldsymbol{\beta}) := Y_{in} - \frac{A_{in}}{M_{ii}} \; . \tag{6}$$

In matrix form we have

$$\tilde{\mathbf{Y}}(\beta) = \mathbf{Y} - (\mathbf{M} \circ \mathbf{I})^{-1}(\mathbf{A}' - [\mathbf{A}'' \;\; \mathbf{A}''\boldsymbol{\beta}]) \; . \tag{7}$$

We stress that (7) is a linear function of $\boldsymbol{\beta}$.

We now need a convex multiclass loss to measure the LOO errors. We could choose the the convex multiclass loss presented in [5], which keeps samples of different classes at the unit marginal distance:

$$L(\boldsymbol{\beta}, i) = \max_{r \neq y_i} |1 + Y_{ir}(\boldsymbol{\beta}) - Y_{iy_i}(\boldsymbol{\beta})|_+, \tag{8}$$

where $|x|_+ := \mathbf{max}(x, 0)$. However, from (1) and (2) it is possible to see that by changing $\boldsymbol{\beta}$ we only change the scores of the target $N+1$-th class. Thus, when using this loss almost all samples are neglected during optimization with respect to $\boldsymbol{\beta}$. We address this issue by proposing a modified version of (8), $L^{\text{mod}}(\boldsymbol{\beta}, i)$ as

$$\begin{cases} |1 + Y_{i(N+1)}(\boldsymbol{\beta}) - Y_{iy_i}(\boldsymbol{\beta})|_+ & : y_i \neq N+1 \\ \max\limits_{r \neq y_i} |1 + Y_{ir}(\boldsymbol{\beta}) - Y_{iy_i}(\boldsymbol{\beta})|_+ & : y_i = N+1 \end{cases}$$

The rationale behind this loss is to enforce a margin of 1 between the target $N+1$-th class and the correct one, even when the $N+1$-th class has not the highest score. This has the advantage of forcing the use of all the samples in the optimization of $\boldsymbol{\beta}$.

Given the LOO errors and the multiclass loss function, we can obtain $\boldsymbol{\beta}$ by solving the convex problem

$$\min_{\boldsymbol{\beta}} \sum_{i=1}^{M} L^{mod}(\boldsymbol{\beta}, i) \qquad \text{s.t.} \;\; \|\boldsymbol{\beta}\|_2 \leq 1, \quad \beta_i \geq 0, i = 1, \ldots, N \; . \tag{9}$$

[1] http://www.idiap.ch/~ikuzbor

Constraining $\boldsymbol{\beta}$ within a unit ℓ_2 ball is a form of regularization on $\boldsymbol{\beta}$, that prevents the overfitting of the parameters $\boldsymbol{\beta}$. This optimization procedure can be implemented elegantly using projected subgradient descent [3], which is not affected by the fact that the objective function in (9) is not differentiable everywhere. The pseudocode of the optimization algorithm is summarized in Alg. 1.

Algorithm 1 Projected subgradient descent to find $\boldsymbol{\beta}$

```
Input: $\mathbf{M}, \mathbf{A}', \mathbf{A}'', T$
Output: $\boldsymbol{\beta}$
 1: $\beta \leftarrow \mathbf{0}$
 2: for $t = 1 \ldots T$ do
 3:     $\tilde{\mathbf{Y}} \leftarrow \mathbf{Y} - (\mathbf{M} \circ \mathbf{I})^{-1}(\mathbf{A}' - [\mathbf{A}'' \;\; \mathbf{A}''\boldsymbol{\beta}])$
 4:     $\Delta \leftarrow \mathbf{0}$
 5:     for $i = 1 \ldots M$ do
 6:         if $y_i \neq N+1$ then
 7:             if $1 + Y_{i(N+1)} - Y_{iy_i} > 0$ then
 8:                 $\Delta \leftarrow \Delta + \frac{\mathbf{A}''_i}{M_{ii}}$
 9:             end if
10:         else if $\mathbf{max}_{r \neq y_i}(1 + Y_{ir} - Y_{iy_i}) > 0$ then
11:             $\Delta \leftarrow \Delta - \frac{\mathbf{A}''_i}{M_{ii}}$
12:         end if
13:     end for
14:     $\boldsymbol{\beta} \leftarrow \boldsymbol{\beta} - \frac{\Delta}{M\sqrt{t}}$
15:     $\beta_i \leftarrow \max(\beta_i, 0) \quad \forall i = 1, \ldots, N$
16:     if $\|\boldsymbol{\beta}\|_2 > 1$ then
17:         $\boldsymbol{\beta} = \frac{\boldsymbol{\beta}}{\|\boldsymbol{\beta}\|_2}$
18:     end if
19: end for
```

Computational complexity for obtaining $\boldsymbol{A}'$, $\boldsymbol{A}''$ and $\mathbf{M}$ is in $\mathcal{O}(M^3 + M^2(N+1))$, which comes from matrix operations (3)-(5). Note that this complexity is better than the one of a classical OVA SVM which in worst case is known to be in $\mathcal{O}(M^3N)$ [10]. The Alg. 1 is in $\mathcal{O}(MN(T+1))$, where we assume that most terms in (7) are precomputed. Each iteration of the algorithm is very efficient since it depends linearly on both training set size and number of classes.

To conclude this section, a compact description of the MULTIpLE algorithm is: i) find the optimal tranfer weights $\boldsymbol{\beta}$ with Alg. 1; ii) calculate the final solution using (1)-(5). The source code of MULTIpLE is available online[1].

5. Experiments

We present here a series of experiments designed to investigate the behavior of our algorithm when (a) the source classes and the target class are related/unrelated, and when (b) the overall number of classes increases. All experiments were conducted on subsets of two different public datasets, and the results were benchmarked against several baselines. In the rest of the section we first describe our experimental setup (section 5.1), then we describe the chosen baselines (section 5.2). Section 5.3 reports our findings.

5.1. Data setup

We run all experiments on subsets of the Caltech-256 database [9] and of the Animal with Attributes (AwA) database [13]. From the Caltech-256 database, we selected a total of 14 classes and for the AwA dataset, 42 classes. We did not carry out any image pre-selection or pre-processing. Moreover, for both databases we used pre-computed features available online[2]. Specifically, for the Caltech-256 experiments we used the following features: oriented and unoriented PHOG shape descriptors, SIFT appearance descriptors, region covariance and local binary patterns totalling in 14 descriptor types [8]. For the AwA experiments the chosen features were SIFT, rgSIFT, SURF, PHOG, RGB color histograms and local self-similarity histograms [13].

For each class considered, we randomly selected 80 image samples. These were then split in three disjoint sets: 30 samples for the source classifier, 20 samples for training and 30 samples for test. The samples of the source classifier were used for training the N models $\boldsymbol{W}'$ (Section 4).

The performance of each method (see Section 5.2) was evaluated using progressively $\{5, 10, 15, 20\}$ training samples for each of the $N+1$ classes. The experiments were repeated 10 times, using different randomly sampled training and test sets, which we refer to as data splits. Furthermore, to get a reliable estimate of the performance of transfer with respect to different classes, we used a leave-one-class-out approach, considering in turn each class as the $N+1$ target class, and the other N as source classifiers. We report results averaged over all data splits and leave-one-class-out evaluations.

5.2. Algorithmic setup

We compared MULTIpLE against two categories of baselines. The first, that we call no transfer baselines, consists of a group of algorithms addressing the $N \longrightarrow N+1$ problem without leveraging over source models; the second, that we call transfer baselines, consists of a group of methods attempting to solve the $N \longrightarrow N+1$ problem by leveraging over source models. The no transfer baselines are the following:

No transfer corresponds to LSSVM trained only on the new training data.

Batch corresponds to a LSSVM trained using all available samples, i.e. assuming to have access to all the data used to build the source models plus the new training data. The performance of this method might be seen as an indicator of the best performance achievable on the problem, thus as an important reference for assessing the results obtained by transfer learning methods.

Source is the LSSVM N-class source classifier. In this case, classification on the sample belonging to $N+1$-th class is assigned 0 accuracy.

Source+1 corresponds to a binary LSSVM trained to discriminate between the target class vs the source classes given the training data. It is evaluated on the $N+1$ problem by combining it with Source in a OVA setting. It is arguably the simplest possible approach to address the $N \longrightarrow N+1$ problem.

Source+1 (hinge) is the scheme analogous to Source+1, but utilizing the hinge loss $\ell(x, z) = |1 - xz|_+$, thus corresponding to a classical SVM formulation.

As transfer baselines, we chose the following methods:

MKTL We compared against Multi Kernel Transfer Learning (MKTL) [11], which is one of the few existing discriminative transfer learning algorithm in multiclass formulation.

MultiKT-OVA We implemented an OVA multiclass extension of the binary transfer learning method by Tommasi *et al.* [18] as follows: as in the standard OVA formulation, we train MultiKT instance to discriminate between one of $N+1$ classes and the rest N. At the same time we use Source as the source classifier. Thus, eventually we obtain $N+1$ MultiKT instances.

PMT-SVM-OVA We also implemented an OVA multiclass extension of the binary transfer learning method by Aytar and Zisserman [1], as done for MultiKT-OVA.

MultisourceTrAdaBoost-OVA As a final transfer learning baseline, we implemented an OVA extension of MultisourceTrAdaBoost [20], where each source corresponds to a subset of samples designated for the source classifier, while belonging to a specific class. We follow the authors by using linear SVM as weak learner.

Apart for PMT-SVM-OVA and MultisourceTrAdaBoost-OVA, which cannot be kernelized, we used all the features available for each dataset via kernel averaging [8], computing the average of RBF kernels over all available features from the dataset at hand and RBF hyperparameters $\gamma \in \{2^{-5}, 2^{-6}, \ldots, 2^{8}\}$. The trade-off hyperparameter $C \in \{10^{-5}, 10^{-6}, \ldots, 10^{8}\}$ was tuned by 5-fold cross-validation for the no transfer baselines. In case of model-transfer algorithms, source model's C value was reused.

Since MultisourceTrAdaBoost-OVA is a non-kernel baseline, to test its performance over multiple features we concatenated them. This approach proved computationally unfeasible for PMT-SVM-OVA (we used the implementation made available by the authors). Thus, to compare fairly with it, we also did run experiments using, for all methods, a linear kernel and a single feature (SIFT for the Caltech-256 and PHOG for the AwA). We refer to this setting as linear.

[2]Caltech-256: `http://www.vision.caltech.edu/Image_Datasets/Caltech256/`
AwA: `http://attributes.kyb.tuebingen.mpg.de/`

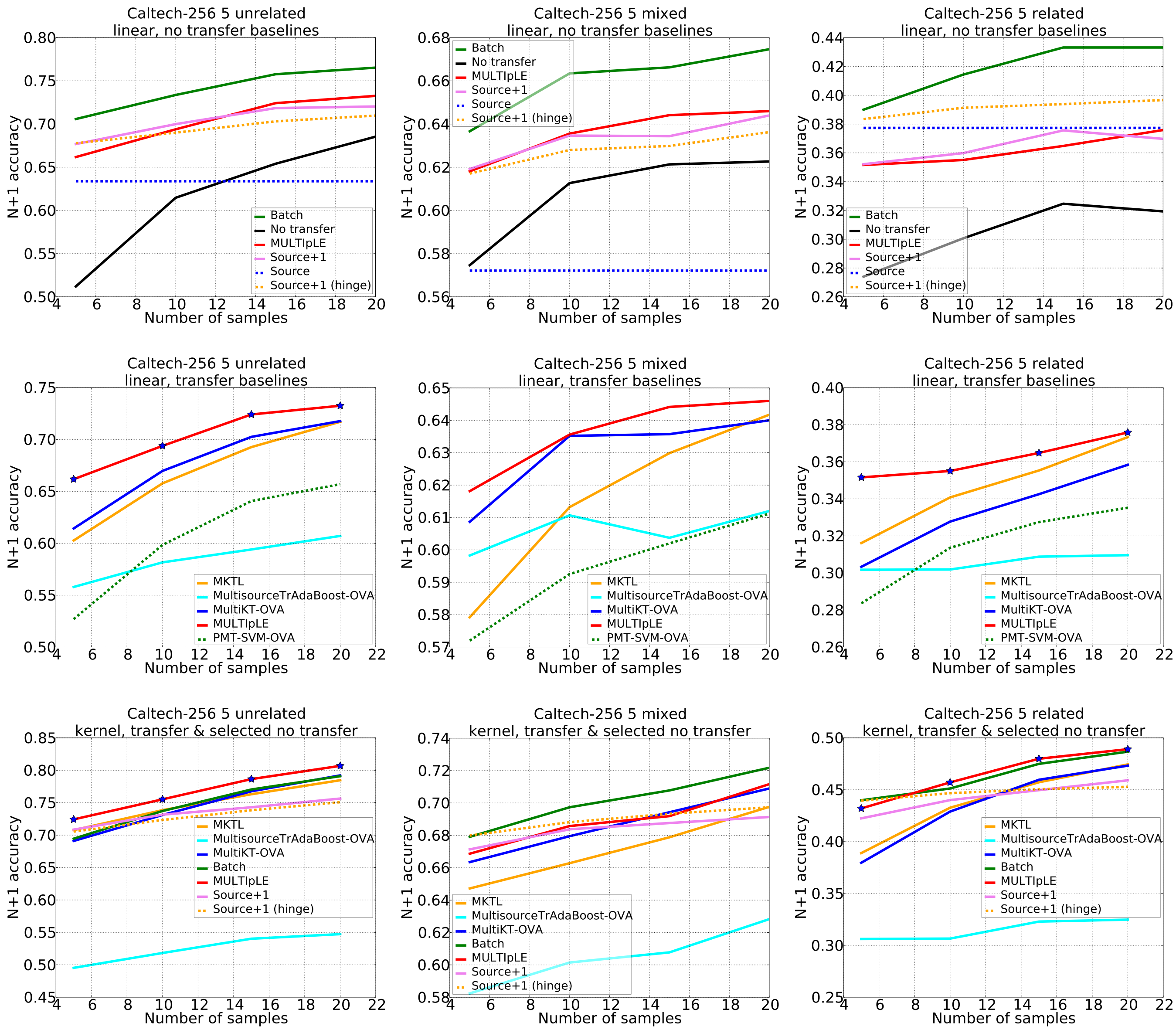

Figure 2: Experimental results for $N + 1 = 5$, Caltech-256. From left to right, columns report results for the unrelated, mixed and related settings. Top row: no transfer baselines, linear case. Middle row: transfer learning baselines, linear case. Bottom row: transfer and competitive no transfer baselines, average of RBF kernels over all features. Stars represent statistical significance of MULTIpLE over MultiKT-OVA, $p < 0.05$.

5.3. Evaluation results

Mimicking the setting proposed in Tommasi *et al.* [18], we performed experiments on different groups of related, unrelated and mixed categories for both databases.

For the Caltech-256 database, the related classes were chosen from the "quadruped animals" subset; the unrelated classes were chosen randomly from the whole dataset, and the mixed classes were taken from the "quadruped animals" and the "ground transportation" subsets, sampled in equal proportions. For the AwA database, the related classes were chosen from the "quadruped animals" subset; the unrelated classes were randomly chosen from the whole dataset, and the mixed classes were sampled in equal proportions from the subsets "quadruped animals" and "aquatic animals". This setting allows us to evaluate how MULTIpLE, and the chosen baselines, are able to exploit the source knowledge in different situations, while considering the overall accuracy. To assess the performance of all methods as the overall number of classes grows, we repeated all experiments in-

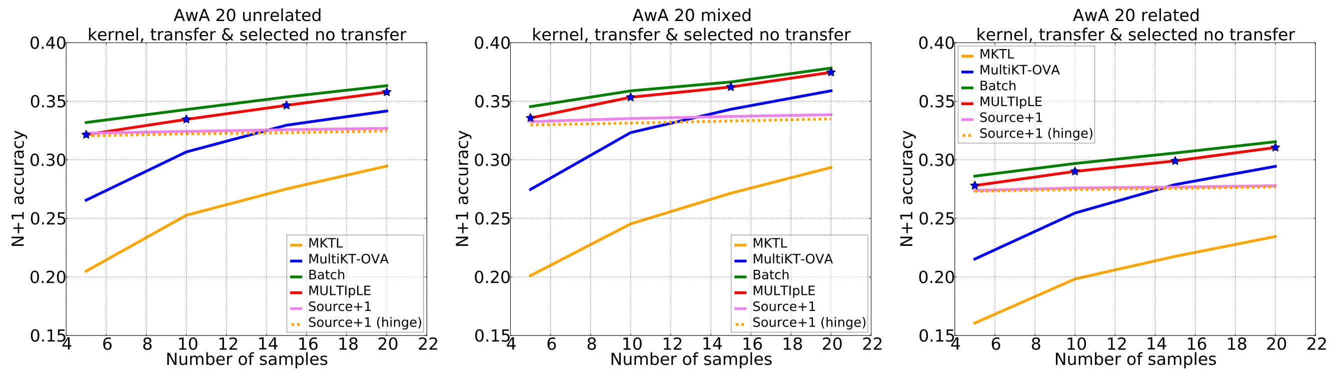

Figure 3: Results for $N + 1 = 20$, AwA, transfer and competitive no transfer baselines, average of RBF kernels, all features. Left to right: unrelated, mixed and related settings. Stars represent statistical significance of MULTIpLE over MultiKT-OVA.

creasing progressively their number, with $N+1 = 5, 10, 20$ respectively. Because of space constraint and redundancy, only a subset of all experiments is reported here[3].

Figure 2 shows the results obtained for $N + 1 = 5$. The left column shows the results for the unrelated setting; the center column shows the results for the mixed setting, and the right column shows the results for the related setting. The first row compares the results obtained by MULTIpLE with those of the no transfer baselines (Section 5.2), using a single feature and a linear kernel. We see that the performance of MULTIpLE is always better than no transfer, and in two cases out of three is better or on par with Source and Source+1 (hinge) (unrelated and mixed), while it is always equivalent to Source+1. This is not the case anymore when using multiple features through kernel averaging (Figure 2, bottom row): when using the kernelized version of all algorithms, our approach always performs equal or better than most baselines, apart for Batch and in rare cases, Source+1 (hinge). Compared to Batch, in two cases out of three (unrelated, related) MULTIpLE performs on par with it. This is a remarkable result, as the Batch method constitutes an important reference for the behavior of transfer learning algorithms in this setting (Section 5.2).

Figure 2, middle row, reports results obtained for MULTIpLE and all transfer learning baselines, as defined in Section 5.2, for one feature and the linear kernel. We see that our algorithm obtains a better performance compared to all the others, especially in the small sample regime. As our method builds on the MultiKT algorithm, we tested the statistical significance of our performance with respect to it, using the Wilcoxon signed-rank test ($p < 0.05$). In two cases out of three (related, unrelated), MULTIpLE is significantly better than its competitor. This is the case also when using all features via kernel averaging. We mark these cases with a star on the plots (Figure 2, middle and bottom row). With respect to the transfer baselines, the related setting seems to be the one more favorable to our approach. With respect to the no transfer baselines, MULTIpLE seems to perform better in the unrelated case.

The performance of PMT-SVM-OVA and Multisource-TrAdaBoost-OVA is disappointing, compared with what achieved by the other two transfer learning baselines, i.e. MultiKT and MKTL. This is true for all settings (related, unrelated and mixed). Particularly, the performance of MultisourceTrAdaBoost-OVA does not seem to benefit from using multiple features (Figure 2, middle and bottom row). On the basis of these results, we did not consider these two baseline algorithms in the rest of our experiments.

Figure 3 shows results for $N + 1 = 20$ classes on the AwA dataset, for the unrelated (left), mixed (center) and related (right) settings, all features (averaged RBF kernels). For sake of readability, we report here only the baselines which were competitive with, or better than, MULTIpLE in the $N + 1 = 5$ case, in at least one setting. We see that here our algorithm consistently outperforms all transfer learning baselines, especially with a small training set, while obtaining a performance remarkably similar to Batch, in terms of accuracy and behavior. The Wilcoxon signed-rank test ($p < 0.05$) indicates that, in all these experiments MULTIpLE is again significantly better than MultiKT-OVA. These results suggest that, as the number of sources grows, our method gets closer to the Batch performance while using only a considerably smaller amount of data – the ultimate goal of any effective transfer learning method. Results obtained on the whole AwA dataset support this claim[3].

6. Discussion and Conclusions

All results confirm our claim that the mere extension to multiclass of existing binary transfer learning algorithms is not sufficient to address the $N \longrightarrow N + 1$ problem. This is well illustrated by the gap in performance between MULTIpLE and MultiKT, which is consistent across datasets,

[3]All experimental results and the source code are available at `http://www.idiap.ch/~ikuzbor`.

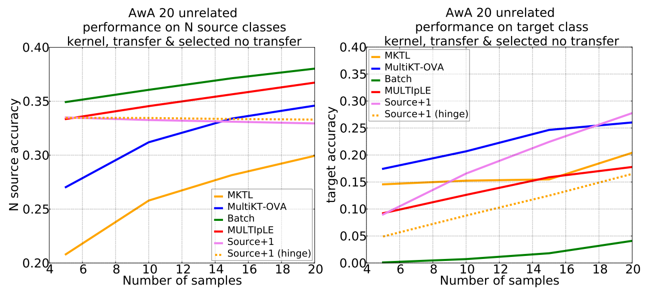

Figure 4: Results for $N+1 = 20$, AwA, unrelated: accuracy over the N sources (left) and over the $+1$ target (right).

settings and the number of classes. The main difference between the two algorithms is the term we added into the objective, that allows to learn the new class, while preserving the performance on the old classes. The results we have shown demonstrate the importance of such a term in the behavior of the algorithm. One might argue that the worse performance of the transfer learning baselines depends on how we implemented the OVA extension for such binary methods. Still, the results obtained by MKTL, the only transfer learning baseline with a multiclass formulation, clearly indicate that the ability to handle multiple sources by itself is not the solution. To gain a better understanding on how MULTIpLE balances the need to preserve performance over the sources, and the learning of the target class, we show the accuracy plots for the AWA experiments, $N + 1 = 20$, unrelated, for the N sources and for the $+1$ target separately (Figure 4). MULTIpLE and Batch present similar behaviors, as they both preserve the accuracy over the N sources. Both methods do not aggressively leverage over sources for learning the target class, as done by MultiKT-OVA and MKTL (to a lesser extent), although MULTIpLE seems to be able to do so better than Batch. Thus, our choice of optimizing the overall accuracy has resulted in a method able to reproduce the behavior and the performance achievable if all training data would be accessible. Note training with all the data might not be possible, nor desirable, in all applications. As opposed to this, the OVA extensions of existing binary transfer learning algorithms are more biased towards a strong exploitation of source knowledge when learning the target class, at the expenses of the overall performance. How to combine these two aspects, namely how to design principled methods able to obtain an overall accuracy comparable to that of the Batch method while at the same time boosting the learning of the target class, remains the open challenge of the $N \longrightarrow N + 1$ transfer learning problem.

7. Acknowledgments

This work is partially supported by the Swiss National Science Foundation Sinergia project Ninapro (I.K. & B.C.). We are thankful to Arjan Gijsberts and Tatiana Tommasi for helping with the coding.

References

[1] Y. Aytar and A. Zisserman. Tabula rasa: Model transfer for object category detection. In *Proc. ICCV*, 2011.

[2] J. Blitzer, R. McDonald, and F. Pereira. Domain adaptation with structural correspondence learning. In *Proc. EMNLP*, pages 120–128. ACL, 2006.

[3] S. Boyd and L. Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[4] G. Cawley. Leave-one-out cross-validation based model selection criteria for weighted LS-SVMs. In *Proc. IJCNN*, 2006.

[5] K. Crammer and Y. Singer. On the algorithmic implementation of multiclass kernel-based vector machines. *JMLR*, 2:265–292, 2002.

[6] H. Daumé. Frustratingly easy domain adaptation. In *Anual meeting, ACL*, 2007.

[7] L. Duan, D. Xu, I. Tsang, and J. Luo. Visual event recognition in videos by learning from web data. *IEEE TPAMI*, 34(9):1667–1680, 2012.

[8] P. Gehler and S. Nowozin. On feature combination for multiclass object classification. In *Proc. ICCV*, 2009.

[9] G. Griffin, A. Holub, and P. Perona. Caltech-256 object category dataset. Technical report, Caltech, 2007.

[10] D. Hush, P. Kelly, C. Scovel, and I. Steinwart. QP algorithms with guaranteed accuracy and run time for support vector machines. *JMLR*, 7:733–769, 2006.

[11] L. Jie, T. Tommasi, and B. Caputo. Multiclass transfer learning from unconstrained priors. In *Proc. ICCV*, pages 1863–1870. IEEE, 2011.

[12] I. Kuzborskij and F. Orabona. Stability and hypothesis transfer learning. In *Proc. ICML*, 2013.

[13] C. Lampert, H. Nickisch, and S. Harmeling. Learning to detect unseen object classes by between-class attribute transfer. In *Proc. CVPR*, pages 951–958. IEEE, 2009.

[14] A. Lazaric, M. Restelli, and A. Bonarini. Transfer of samples in batch reinforcement learning. In *Proc. ICML*, 2008.

[15] Y. Mansour, M. Mohri, and A. Rostamizadeh. Domain adaptation with multiple sources. *Proc. NIPS*, 21:1041–1048, 2009.

[16] J. Suykens and J. Vandewalle. Least squares support vector machine classifiers. *Neural Process. Lett.*, 9(3), 1999.

[17] S. Thrun. Is learning the n-th thing any easier than learning the first? In *Proc. NIPS*, pages 640–646, 1995.

[18] T. Tommasi, F. Orabona, and B. Caputo. Safety in numbers: Learning categories from few examples with multi model knowledge transfer. In *Proc. CVPR*, pages 3081–3088. IEEE, 2010.

[19] J. Yang, R. Yan, and A. Hauptmann. Cross-domain video concept detection using adaptive SVMs. In *Proc. ACM*, pages 188–197. ACM, 2007.

[20] Y. Yao and G. Doretto. Boosting for transfer learning with multiple sources. In *Proc. CVPR*, 2010.