Domain Adaptation for Sequential Detection

Šimon Fojtů, Karel Zimmermann, Tomáš Pajdla, and Václav Hlaváč

Center for Machine Perception
Czech Technical University in Prague
Technická 2, Praha 6
fojtusim@cmp.felk.cvut.cz

Abstract. We propose a domain adaptation method for sequential decision-making process. While most of the state-of-the-art approaches focus on SVM detectors, we propose the domain adaptation method for the sequential detector similar to WaldBoost, which is suitable for real-time processing. The work is motivated by applications in surveillance, where detectors must be adapted to new observation conditions. We address the situation, where the new observation is related to the previous observation by a parametric transformation. We propose a learning procedure, which reveals the hidden transformation between the old and new data. The transformation essentially allows to transfer the knowledge from the old data to the new one. We show that our method can achieve a 60% speedup in the training w.r.t. the baseline WaldBoost algorithm while outperforming it in precision.

1 Introduction

Object detectors need to run under different observation conditions (e.g. view angle or illumination) in autonomous surveillance applications. Possible variations of the observation conditions and corresponding objects appearance are usually not known in advance. When observation conditions change, the performance of the detector decreases. Retraining the detector on the new training data is inconvenient, since manual annotations are extremely time consuming and supervised learning of the accurate object detector often requires a huge amount of training data [1].

To avoid the necessity of capturing and annotating a huge training dataset every time under new observation conditions, we propose to learn a new detector from (i) a small number of new data and (ii) from a collection of previous datasets captured under similar observation conditions, or a previously learned detector. By utilizing this *prior knowledge*, the number of training samples can be significantly reduced.

We aim at reducing the number of training samples required for a training of a multiview classifier by introducing a transformation between the views. The effect of using the transformation during the training is threefold. First, it constrains the feature selection process. In the case of Haar-like features, the space of all possible features is overcomplete, thus the reduction is significant. Second, it helps to prevent overfitting, as the transformed samples work as regularizers in fact. Third, the number of samples is effectively summed, as the training process takes information from all the views through the transformation.

J.-K. Kämäräinen and M. Koskela (Eds.): SCIA 2013, LNCS 7944, pp. 215–224, 2013.

Contribution The vast majority of the State-of-the-Art approaches consider the previous dataset unknown and adapt only the previous classifier to the new dataset. In contrast to this, we propose to preserve the original knowledge and train completely new classifier from the previous and new data simultaneously. Furthermore, whereas most of the works on domain adaptation are restricted in the SVM classifier domain, we propose the domain adaptation for the sequential detection (e.g. WaldBoost [2]), which is suitable for real-time processing thanks to the sequential decision-making process. We also make our dataset publicly available for comparison purposes.[1] We propose to exploit the visual similarity of objects from different views by constraining the features used for the classification.

The comparison to the other state of the art methods is given in the Section 2.

2 Related Work

There are several approaches trying to tackle the problem of reducing the number of training samples. This section aims to give their brief overview.

2.1 SVM Based Classifiers

Tommasi et al. [3,4] use the prior least-squares SVM classifiers as regularizers, so that the derived classifiers' hyperplanes are close to the original hyperplanes. This strategy is unfortunately not applicable on cascaded classifiers due to the fact, that the cascade is built incrementally, with later stages focusing on harder samples.

Another approach is used by Jie et al. [5], who add the outputs of previously trained classifiers as another features. This inevitably leads to higher processing time.

Aytar and Zissermann [6] present several modifications to standard SVM allowing adaptation to the target domain. The similarity to our work lies in the introduction of a deformable template of Histograms of Oriented Gradients (HOG).

2.2 Cascaded Classifiers

In the work of Cao et al. [7], a cascaded classifier is presented, in which each stage can either accept, reject, or pass a sample to the next stage. Once the classifier is trained on some dataset, it can be retrained using a small number of samples from a target dataset using Cross Entropy method by sampling new thresholds around the original thresholds.

The following two approaches are concerned with online adaptation of classifiers. First, the work of Grabner et al. [8] enables the on-line adaptation of WaldBoost classifier cascade based on Wald's sequential decision [9]. They introduce a pool of selectors and each selector holds a pool of weak classifiers. The on-line boosting is then performed on the selectors rather than on the weak classifiers in order to adapt the strong classifier complexity. Each training sample is first used as a testing sample to update Wald statistics and then used to adapt the strong classifier.

[1] `cmp.felk.cvut.cz/~fojtusim/kt`

Second, Visentini et al. [10] propose an algorithm for cascaded online boosting. They construct a classifier for each image separately from an ensemble of weak classifiers in real–time. They employ weak classifiers with various features (Haar, Local Binary Patterns, Colour histograms). The shape of the cascade (the number of stages and the number of classifiers in each stage) is automatically adjusted to preserve the real–time constraint.

2.3 Others

Another approach to adaptation of a classifier to a new domain is presented by Jain and Learned-Miller [11]. They propose to smooth the classification boundary, which is based on the assumption that similar samples should have similar classification scores. A Gaussian Process Regression is used for the decision boundary smoothing, i.e. updating classification scores of samples near the boundary.

To the best of our knowledge, there is no other approach trying to adapt a sequential decision process classifier, i.e. WaldBoost–type cascaded classifier. The most similar approach is the On-line WaldBoost [8] by Grabner et al. But in their case, the adaptation is based only on selection of the best weak classifiers from a fixed pool, whereas we try to actively optimize each weak classifier.

3 Method

Given two images of an object, each capturing it from a different but similar viewpoint, there exists a transformation between non-occluded pixels in the images. We assume that there also exists a transformation mapping features from one image to the other. We aim to exploit this fact by learning this transformation during the training of the classifier. The transformation can be modeled arbitrarily in general, but we constrain the transformation to be a similarity transform in this paper. This means that the transformation can change the position and scale of the Haar features, but the type and region weights remain unchanged. The proposed approach is not limited to two views, we can consider one view as a source and look for transformations to N different views, as is shown in Figure 1. Moreover, we treat the features in each stage of the cascaded classifier as independent, which significantly simplifies the problem at the cost of suboptimal performance. Real-time detection of objects is enabled thanks to the employed cascaded classifier structure and the use of Haar-like features in combination with integral images [12].

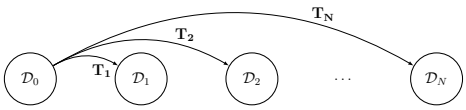


Fig. 1. Transformation of features between the datasets

Let us now introduce the required notation. Let $\mathcal{F}$ denote a Haar-like feature. This feature is determined by its position $\mathbf{p}$, size $\mathbf{s}$, and type t (rectangles and weights). Feature $\mathcal{F}$ maps an image x to a real number $\mathcal{F}_{\mathbf{p},\mathbf{s},t} : \mathbf{x} \to \mathcal{R}$. Let $\mathcal{T}$ be a transformation of a feature $\mathcal{T} : \mathcal{F} \to \mathcal{F}$.

The transformation applied to a feature must be feasible, which means that the transformed feature $\mathcal{F}' = T(\mathcal{F})$ must lie within the image borders. We may also imply other restrictions on the transformation, such that the difference in position is smaller than θ_t and the change in scale smaller than θ_s. If the constraints θ_t and θ_s are set to 0, the transformation is identity. In the other extreme, the constraints values can be so high, that the only limit is the image size. In the first case, in which T becomes identity, the feature is the same in all views. This is equivalent to training a classifier from a joint of all the views. On the other hand, if we do not restrict the transformation (i.e. the only constraint is its feasibility), then there is no connection between the data in the views. The results are equal to the training of separate classifiers, each using data only from one view. The value of the θ_s, θ_t parameters directly influences the training time and to some extent the precision of the final classifier. The higher the value, the more features are explored during the training, which increases the precision but on the other hand increases the time requirements. The introduction of constraints θ_s and θ_t reduces the search space significantly. With discretized position values (pixels) and scales we are able to perform an exhaustive search over all feasible transformations of the original feature.

The remaining question is *how to determine the transformation*?

We propose two methods (Sections 3.1 and 3.2) to deal with this question. The first, called *multiview* method, attempts to train both the weak classifiers and the transformation from scratch simultaneously. The second one named *incremental* uses a prior classifier and builds on it. The details of both are covered in the following sections.

Let us now describe the baseline algorithm, to which the performance of our methods is compared. This training algorithm is also used to train prior classifiers.

The classifier structure is a cascade of weak classifiers, trained using the WaldBoost [2] algorithm. Each stage of the cascade, i.e. weak classifier, consists of a single regression stump function associated with a Haar-like feature $\mathcal{F}$. The error of a weak classifier is given by the sum of weights of misclassified samples

$$e(w, y, x, \mathcal{F}) = \sum_{i=1}^{m} w_i \cdot \mathbf{1}\left(y_i \cdot \operatorname{sign}\left(h(\mathcal{F}(x_i))\right) < 0\right), \tag{1}$$

where m is the number of samples, w is the weight of a sample, y its correct label, $h(x)$ the response of the weak classifier, and $\mathbf{1}$ is the indicator function.

The regression stump function $h(x)$ splits the feature descriptor range into several *bins*. The value of the function $h(x)$ in bin i is computed as a weighted mean of training samples falling into the bin i

$$h_i(x, \mathcal{F}) = \frac{1}{|X_i|} \sum_{j:\mathcal{F}(x_j)\in X_i} \frac{w_j \cdot y_j}{w_j}, \tag{2}$$

where X_i are samples falling into bin i, w_j and y_j are the weight and true label of the j^{th} sample. The regression stump function is often effectively implemented as a look-up table to speed up the classification.

Moreover, we utilize the bootstrap method to extract negative training samples from background images by applying the classifier on those images and adding the false positives to the training set. In this way, the memory requirements of the training algorithm can be controlled.

The more stages of the cascade are evaluated, the more time it takes to decide, whether the sample belongs to the class. An early rejection mechanism is employed [13], in order to speed up the process of classification. Negative training samples that are early rejected are also dropped from the training set and are replaced by newly bootstrapped samples. In this way, each stage is trained on harder samples than the previous one, since the simple samples are already discarded. We do not consider the early acceptance of samples, since in the situation of the rare event detection, the speedup brought about by accepting the positive samples earlier than at the last stage is negligible.

3.1 Multiview Training

The first proposed method requires no prior knowledge. The weak classifiers and the transformations to the views are determined together without any further information. A new pool of n Haar-like features is generated for the training of each stage. A transformation from the feature pool to each view is determined by boosting. The optimal feature and transformations are found together.

The optimal feature is given by

$$\mathcal{F}^* = \arg\min_{\mathcal{F}} \left[\max_{k=1}^{N} \sum_{i:x_i \in X_k} e(w_i, y_i, x_i, T(\mathcal{F})) \right], \tag{3}$$

where X_k denotes data from the k^{th} view. This equation takes into account the correct classification of samples in each view since the optimal transformation to view k, denoted by T_k^* is given by

$$T_k^* = \arg\min_{T} \sum_{i:x_i \in X_k} e(w_i, y_i, x_i, T(\mathcal{F})). \tag{4}$$

For each feature from the pool, all feasible transformations are tested and the one with the lowest error is remembered. The feature that minimizes the maximum weak classifier error over all views is selected. The optimal transformation T_k^* is determined during training for every stage separately.

3.2 Incremental Training

Unlike the multiview method, the incremental method takes as the input also a previously trained classifier, referred to as the *prior*. In this case, the training is done as follows. We do not generate a pool of Haar-like features, but instead during training

(a) View I (b) View II (c) View III (d) View IV

Fig. 2. Examples from the dataset. The dataset contains images of vehicles from four different viewpoints.

of each stage, we take the corresponding feature from the prior classifier and exhaustively search the feature space as in the first method. The effect of the prior classifier is twofold. It again significantly reduces the search space and also guides the direction by restricting the features to a specific type and approximate position.

On a side note, this approach can be also used for retraining classifiers in the same domain, as we can use the classical random feature sampling as a first step and fine-tune the features in a second training run. The final classifier is better than the prior as expected and the time required for the training is much less than if we tried to search the whole feature space exhaustively in the first place.

The parameters of the regression stump functions and the early rejection thresholds are determined from the training and validation data, respectively. No restriction is imposed on them.

4 Experiments

The experimental dataset consists of four sets, each one capturing the vehicles from a slightly different angle. Figure 2 shows samples from each of the views. There are 214, 172, 155, and 106 annotated cars in the respective views. The images were captured on a busy street in the heart of Prague. Although the camera was fixed in each view, the annotated vehicles in one view are not all captured from the same angle, so even in one view, there is some variability in the data. For the negative samples, there are 243 negative backgrounds—photos from an urban environment that either do not contain vehicles or only vehicles that are severely occluded (e.g. only less than 20% of the vehicle is visible). In the experiments only data from view I and II are used. This choice has no significant influence on the performance, but other combinations are not shown in this paper due to size limitation.

We have performed three different experiments to support the validity of our assumptions and ideas and to show the increase in performance compared to classically trained classifiers. The first experiment is concerned with the effects of multiview training on two views. The second experiment shows the improvement brought about by applying the incremental training on a classifier and evaluating both of them on the same data. The last experiment evaluates the improvement of the classifier performance with respect to a varying number of training samples from the target dataset.

4.1 Multiview Training

The first experiment shows the simultaneous training of two classifiers on two views I and II from the dataset. We train 5 different classifiers denoted *Multiview I* and *II*, *Single view I* and *II* and *Merged*. Classifiers *Single view I* and *II* are trained on view I and II, respectively, using the baseline algorithm. Classifier *Merged* is trained on the joint of views I and II, using the baseline algorithm as well. Classifiers *Multiview I* and *II* are trained using the multiview training method (feature transformation) on view I and II, respectively.

The transformation $T(\cdot)$ is constrained to be *close* to identity, in particular, we set $\theta_t = 5$ and $\theta_s = 0.1$. This forces the selected features to be similar to each other and also speeds up the training process, as already discussed.

In order to fairly compare the performance of the classifiers, they are shortened to the same length (12 stages, which is the length of the *Multiview* classifiers). We have plotted three receiver operating characteristic (ROC) curves, each for one dataset (view I, view II, and view I $\cup$ II). These are shown in Figures 3a, 3b, and 3c. The performance of the multiview classifiers outperforms both the baseline classifiers trained on a single view and on the merged views. Note, that the *merged* classifier is trained on twice as much positive samples as the single view classifiers.

4.2 Incremental Training

In this experiment we explore the incremental training. We have two datasets, prior (View I) and target (View II). We want to show the effect of the transformation on the performance and also show the reduction of the time required for the training. We train three classifiers. Classifier *Prior* and *Target* are trained on the prior and target datasets, respectively, using the baseline algorithm. Classifier *Incremental* is trained using the incremental method given the *Prior* classifier and data from target dataset.

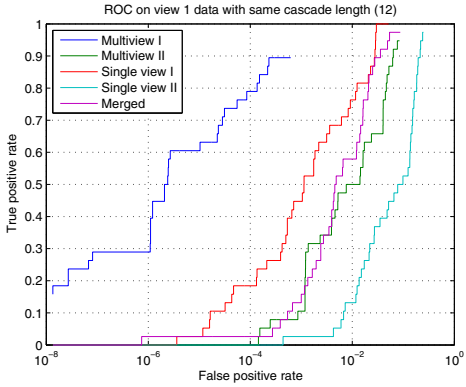
This experiment introduces two improvements. First, the performance of the *Incremental* classifier on the target dataset outperforms the *Target* classifier. Second, the time required to train the *Incremental* classifier is reduced.

Figure 4a shows the performance of all three classifiers on the target dataset. The ROC curve of the prior classifier is shown to emphasize the boost in performance of the incremental classifier (This is more thoroughly explored in Section 4.3). Note its poor performance, which is expected and shows that the two datasets are different. More important is the increase in performance of the *Incremental* classifier w.r.t. the *Target* classifier. The decrease in time, required to train the incremental classifier is shown in Table 1. The time requirements drop to less than 60% of the baseline algorithm.

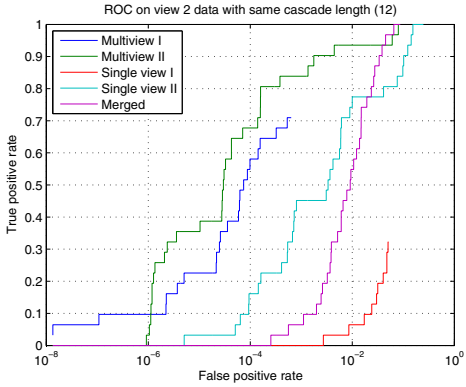
4.3 Incremental Training with Variable Number of Training Samples

We train several classifiers based on a single prior, in order to quantitatively evaluate the benefits of incremental training.

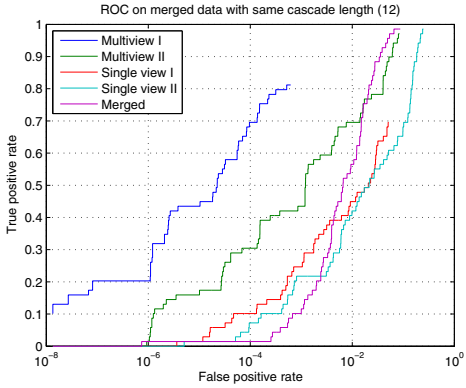
First, a *Prior* classifier is trained on the prior dataset (View I) using the baseline WaldBoost algorithm. Similarly a *Target* classifier is trained on the target dataset (View II), using the same method.



(a) ROC on View I



(b) ROC on View II

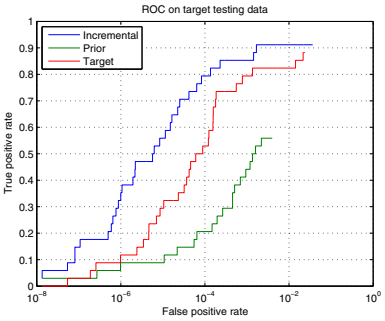


(c) ROC on View I∪II

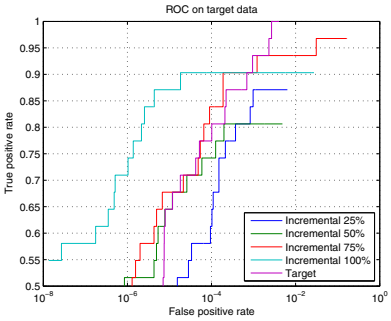
Fig. 3. Comparison of baseline classifiers (*Single View I, II* and *Merged*) on datasets (view I, view II, and view I ∪ II) versus the proposed approach (*Multiview I, II*)

Table 1. Comparison of time required to train classifiers using the baseline, incremental and multiview methods (times averaged over 5 independent trainings) on a Intel i5-2520M CPU, 2.50GHz computer

classifier	mean training time
baseline	877 ± 231 s
incremental	$\mathbf{523 \pm 203}$ **s**
multiview	12504 ± 4022 s



(a) ROC curve comparing performance of classifier *Target* trained on target dataset and classifier *Incremental* trained on target dataset given a *Prior* classifier.



(b) ROC curve comparing performance of a classifier trained on target dataset only (*Target*) and performance of incrementally trained classifiers on subset of the target dataset (*Incremental* 25%, 50%, 75%, and 100%).

Fig. 4.

Second, four subsets of the target dataset are randomly selected (25, 50, 75, and 100%) and four *Incremental* classifiers are trained given the *prior* classifier and one of the subsets using the incremental method.

The *Target* and *Incremental* classifiers are compared against each other on the whole target dataset. The obtained results are shown in Figure 4b. The precision of the *Target* classifier is comparable to the precision of the *Incremental* classifiers trained on 50% and 75% of the target dataset. This shows that it is sufficient to use only half of the target dataset to achieve comparable precision. The performance of the classifier trained only on 25% of the data is lower, but the difference is not very significant. On the other hand, the performance of the *Incremental* classifier trained on the full target dataset outperforms the *Target* classifier by an order of magnitude.

5 Conclusion

We present our approach to adaptation of a sequential decision process–cascaded classifier based on the WaldBoost algorithm–to a target dataset. The proposed idea is to introduce a transformation between Haar-like features. In this paper, the transformation

is restricted to be a similarity, but nevertheless the results show, that the idea is sound and valid. Through the experiments we have shown that by constraining the features using the transformation, we can achieve a speedup of about 160% in the training while outperforming the baseline WaldBoost algorithm in precision. The multiview training method is shown to outperform both classifiers trained on separate views (I and II) and also on the joint of the data. Moreover, our method can be used to retrain a classifier on the same data to increase its performance.

5.1 Future Work

First, the transformation can be relaxed from a similarity. For example, an elastic deformation could better capture the relation between two views. Moreover, the transformation should be constant throughout the cascade in principle, since the views are also constant. Second, more experiments will be performed in order to test the performance of the incremental method with very low number of training samples to explore the field of one-shot learning.

Acknowledgements. Šimon Fojtů was supported by SGS12/187/OHK3/3T/13 project, Karel Zimmermann by P103/11/P700 project, Tomáš Pajdla by TA01031478 AUTMODO project, and Václav Hlaváč by FP7-ICT-247870 NIFTI project.

References

1. Huang, C., Ai, H., Li, Y., Lao, S.: High-performance rotation invariant multiview face detection. PAMI 29(4), 671–686 (2007)
2. Šochman, J., Matas, J.: Waldboost-learning for time constrained sequential detection. In: CVPR (2005)
3. Tommasi, T., Caputo, B.: The more you know, the less you learn: from knowledge transfer to one-shot learning of object categories. In: BMVC, pp. 1–11 (2009)
4. Tommasi, T., Orabona, F., Caputo, B.: Safety in numbers: Learning categories from few examples with multi model knowledge transfer. In: CVPR, pp. 3081–3088 (2010)
5. Jie, L., Tommasi, T., Caputo, B.: Multiclass transfer learning from unconstrained priors. In: ICCV (2011)
6. Aytar, Y., Zisserman, A.: Tabula rasa: Model transfer for object category detection. In: ICCV, pp. 2252–2259 (2011)
7. Cao, X., Wang, Z., Yan, P., Li, X.: Rapid pedestrian detection in unseen scenes. Neurocomputing 74(17), 3343–3350 (2011)
8. Grabner, H., Šochman, J., Bischof, H., Matas, J.: Training sequential on-line boosting classifier for visual tracking. In: ICPR, pp. 1–4 (2008)
9. Wald, A.: Sequential Analysis. Dover Phoenix Editions. Dover Publications (2004)
10. Visentini, I., Snidaro, L., Foresti, G.L.: Cascaded online boosting. Journal of Real-Time Image Processing 5(4), 245–257 (2010)
11. Jain, V., Learned-Miller, E.: Online domain adaptation of a pre-trained cascade of classifiers. In: CVPR, pp. 577–584 (2011)
12. Viola, P., Jones, M.: Rapid object detection using a boosted cascade of simple features. In: CVPR (2001)
13. Freund, Y., Schapire, R.E.: A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting. Journal of Computer and System Sciences 55(1), 119–139 (1997)