Taylor Expansion Based Classifier Adaptation: Application to Person Detection

Cha Zhang[†], Raffay Hamid[‡] and Zhengyou Zhang[†]
[†]Microsoft Research, One Microsoft Way, Redmond, WA 98052
[‡]Georgia Institute of Technology, Atlanta, GA 30332
[†]{chazhang,zhang}@microsoft.com, [‡]raffay@cc.gatech.edu

Abstract

Because of the large variation across different environments, a generic classifier trained on extensive data-sets may perform sub-optimally in a particular test environment. In this paper, we present a general framework for classifier adaptation, which improves an existing generic classifier in the new test environment. Viewing classifier learning as a cost minimization problem, we perform classifier adaptation by combining the cost function on the old data-sets with the cost function on the data-set collected from the new environment. The former term is further approximated with its second order Taylor expansion to reduce the amount of information that needs to be saved for adaptation. Unlike traditional approaches that are often designed for a specific application or classifier, our scheme is applicable to various types of classifiers and user labels. We demonstrate this property on two popular classifiers (logistic regression and boosting), while using two types of user labels (direct labels and similarity labels). Extensive experiments conducted for the task of person detection in conference-room environments show that significant performance improvement can be achieved with our proposed method.

1. Introduction

Pattern classification has been one of the most important tasks in computer vision. In the past decades, learning based data-driven methods have demonstrated superior performance on various classification tasks, such as object detection, object recognition, tracking, *etc*. The performance of a learning based classifier depends heavily on the representativeness of the labeled data used during training. If the training data contains only a small number of examples sampled in a particular test environment, the learned classifier may be too specific to be generalized to unseen data. On the other hand, if the training data is extensive, the classifier may generalize well but perform sub-optimally in a particular test environment.

In many situations, a generic classifier is needed to perform detection task for a variety of environments. For instance, when a person detector is applied in users' offices to detect people and set their presence status automatically, a generic detector is necessary so that the application can work in any office. On the other hand, although the test environment is unknown, the variation of the test environment is generally limited *after* the deployment of the classifier. That is, once the person detector has been deployed in the end user's office, the color of the walls, the furniture, the lighting condition, and even the people seen in the office remain largely the same. Ideally, we would like to have a mechanism for the generic classifier to adapt itself to the new environment and improve its performance over time.

In this paper, we study the problem of *classifier adaptation*, namely, how to adapt a generic classifier trained from extensive data-sets and improve its performance in a particular test environment. There are two main challenges in classifier adaptation. Firstly, additional labeled data collected in the new test environment are often required for the adaptation. Since labeling is an expensive task for the end user, the amount of additional data has to be minimal. This is also the main obstacle to training a classifier from scratch only for that particular test environment. Secondly, a typical generic classifier may require thousands of example images or billions of sub-windows for training (*e.g.*, the face detector in [22]). It is usually impractical to send these data to the end user for a batch retraining of the classifier. One has to limit the amount of data associated with the generic classifier to be sent to the end user for classifier adaptation.

In various recognition tasks such as speech recognition and handwriting recognition, adaptation has become an indispensable tool to achieve high recognition accuracy. However, the importance of classifier adaptation for object detection tasks has rarely been explored in literature. Recent developments of adaptation algorithms for tracking [4, 1, 14], in particular online boosting based methods [11, 15], can be viewed as efforts toward classifier adaptation, but few have directly addressed the problem of adapting an *existing* generic detector to a particular test environment. In addition, most existing adaptation schemes

978-1-4244-2243-2/08/$25.00 ©2008 IEEE

are designed for a particular application, and applicable to a particular type of classifier and a particular type of user label. As a result, few can be easily applied to different applications with different classifiers or user labels.

The main contribution of this paper is two-fold. First, we present a general formulation of the classifier adaptation problem, and a novel Taylor expansion based adaptation method that is applicable to many classifiers. With our proposed method, the amount of data to be sent to the end user is only the gradient and Hessian of the classifier, which is typically very small. We demonstrate the application of the proposed method on two popular machine learning algorithms – logistic regression and boosting. Second, in addition to the typical direct user labels which are given to individual examples, we introduce adaptation based on similarity labels, which are given to pairs of examples indicating whether they have the same, although unknown, labels. Similarity labels can be derived by an automatic tracking algorithm in video sequences, which creates a novel unsupervised classifier adaptation scheme that can be complementary to existing direct label based co-training schemes such as [13, 18, 12]. Experiments are conducted on a challenging person detection task in conference rooms. We show that significant performance improvement can be achieved after adaptation from a generic person detector.

The paper is organized as follows. Section 2 presents a general formulation of the detection adaptation problem and the Taylor expansion based solution. The proposed algorithm is applied to logistic regression and boosting classifiers in Section 3, along with the two forms of example labels. In Section 4 we discuss the relationship of the proposed method with a number of existing approaches in the literature. Experiments and conclusions are given in Section 5 and 6, respectively.

2. Problem Formulation

2.1. Parametric Learning

Without loss of generality, consider a two-class classification problem as follows. A set of labeled examples $\mathcal{S} = \{(x_k, t_k), k = 1, \cdots, K\}$ are given for training, where $t_k = 1$ for positive examples and $t_k = 0$ for negative examples. A parametric learning algorithm intends to find a mapping function

$$y = F(x|\mathbf{\Theta}), \tag{1}$$

where y is the predicted label of the example, and $\mathbf{\Theta}$ is a set of parameters for the mapping function. In order to find the optimal mapping parameters, a common practice is to define a cost function

$$C(F(x|\mathbf{\Theta}), \mathcal{S}) \tag{2}$$

over the training data-set. Various optimization algorithms can then be applied to minimize the cost function in order to obtain the optimal parameters $\mathbf{\Theta}$.

2.2. Detector Adaptation

During adaptation, a generic classifier trained on an extensive data-set $\mathcal{S}^{(o)}$ is given, denoted as $F(x|\mathbf{\Theta}^{(o)})$, where superscript $^{(o)}$ indicates that the parameters are optimal on the *old* data. The user has collected data $\mathcal{S}^{(n)}$ from a new test environment, where superscript $^{(n)}$ indicates *new*. The goal is to find a new set of parameters $\mathbf{\Theta}^{(n)}$, such that the classifier performs better in the new test environment.

While there are many possible ways to find $\mathbf{\Theta}^{(n)}$, in this paper we propose to use:

$$\begin{aligned} \mathbf{\Theta}^{(n)} &= \arg\min_{\mathbf{\Theta}} J(\mathbf{\Theta}) \\ &= \arg\min_{\mathbf{\Theta}} C(F(x|\mathbf{\Theta}), \mathcal{S}^{(o)}) + \lambda D(F(x|\mathbf{\Theta}), \mathcal{S}^{(n)}), \end{aligned} \tag{3}$$

where J is the revised overall cost function for adaptation; D is a cost function defined on the new data-set; and λ is a parameter controlling the relative importance of the old and the new data-set. Note D may be different from C because the labels on the new data-set may be in a different form (*e.g.*, Section 3.2.2).

Eq. (3) combines the cost functions on the old and the new data-sets, which ensures that the adapted classifier can work well even when the new data-set is very small. The remaining challenge is how to obtain $C(F(x|\mathbf{\Theta}), \mathcal{S}^{(o)})$. As mentioned earlier, the old data-set $\mathcal{S}^{(o)}$ is extensive and too huge to be made available for adaptation which is performed at the end user's side. Our proposal is to use a compact representation or an approximation of the cost function on the old data-set to replace $C(F(x|\mathbf{\Theta}), \mathcal{S}^{(o)})$ in Eq. (3) during adaptation, *i.e.*:

$$C(F(x|\mathbf{\Theta}), \mathcal{S}^{(o)}) \approx \tilde{C}(F(x|\mathbf{\Theta}), \mathbb{C}(\mathcal{S}^{(o)})), \tag{4}$$

where $\mathbb{C}(\mathcal{S}^{(o)})$ is a compact representation of the old data-set $\mathcal{S}^{(o)}$. Depending on the classifier's formulation, such a compact representation may or may not be easy to find. Below we present a widely applicable approximation method based on Taylor expansion.

2.3. Taylor Expansion Based Adaptation

We propose to use the Taylor expansion of the cost function on the old data as an approximation. For instance, with the second order Taylor expansion at the previously trained parameters $\mathbf{\Theta}^{(o)}$, we have:

$$\begin{aligned} C(F(x|\mathbf{\Theta})) \approx\ & C(F(x|\mathbf{\Theta}^{(o)})) + \\ & \nabla C(F(x|\mathbf{\Theta}^{(o)}))(\mathbf{\Theta} - \mathbf{\Theta}^{(o)}) + \\ & \frac{1}{2}(\mathbf{\Theta} - \mathbf{\Theta}^{(o)})^T \mathbf{H}_C(\mathbf{\Theta}^{(o)})(\mathbf{\Theta} - \mathbf{\Theta}^{(o)}), \end{aligned} \tag{5}$$

where we represent parameter $\mathbf{\Theta}$ in a vector form and omit symbol $\mathcal{S}^{(o)}$ for conciseness. $\nabla C(F(x|\mathbf{\Theta}^{(o)}))$ is the gradient of the cost function, and $\mathbf{H}_C(\mathbf{\Theta}^{(o)})$ is the Hessian matrix whose elements comprise the second order derivative of the cost function with respect to $\mathbf{\Theta}$. With this approximation, the adaptation algorithm only needs to receive $\nabla C(F(x|\mathbf{\Theta}^{(o)}))$ and $\mathbf{H}_C(\mathbf{\Theta}^{(o)})$, which are generally much smaller in size compared with the original data-set $\mathcal{S}^{(o)}$.

Note the above Taylor expansion approximation is valid for smooth multivariate functions where $\nabla C(F(x|\mathbf{\Theta}))$ and $\mathbf{H}_C(\mathbf{\Theta})$ exist within a *ball* in the space of $\mathbf{\Theta}$ with center at $\mathbf{\Theta}^{(o)}$ [10]. The error of the approximation is on the order of $||\mathbf{\Theta} - \mathbf{\Theta}^{(o)}||^3$. In practice, we note many cost functions of parametric machine learning algorithms are indeed smooth around the optimized $\mathbf{\Theta}^{(o)}$. In the following, we demonstrate the application of the above adaptation scheme on two popular machine learning algorithms: logistic regression and boosting.

3. Adaptation of Logistic Regression Classifiers and Boosting Classifiers

3.1. Logistic Regression

Logistic regression is a very popular tool in machine learning [2]. In this method, a set of features $f_j(\cdot), j = 1, \cdots, J$ are extracted from the training data set $\mathcal{S} = \{(x_k, t_k), k = 1, \cdots, K\}$. The likelihood of an example x_k being a positive example is:

$$p_k = \frac{1}{1 + \exp\{-\sum_j w_j f_j(x_k)\}}, \tag{6}$$

where w_j is the set of parameters to be determined. The likelihood function of the whole data-set can be written as:

$$P = \prod_k p_k^{t_k} (1 - p_k)^{1 - t_k}. \tag{7}$$

As usual, we can define a cost function by taking the negative logarithm of the likelihood, which gives the *cross-entropy* error function as:

$$C \triangleq -\frac{1}{K} \ln P = -\frac{1}{K} \sum_k \{t_k \ln p_k + (1 - t_k) \ln(1 - p_k)\}. \tag{8}$$

Logistic regression minimizes the above cost function on the training data-set to find the optimal set of parameters w_j. We refer the readers to [17] for a comparison between various algorithms to solve logistic regression.

3.2. Adaptation of Logistic Regression Classifier

The gradient and Hessian of the logistic regression error function with respect to the parameters w_j can be easily computed as [2]:

$$\frac{\partial C}{\partial w_j} = \frac{1}{K} \sum_k (p_k - t_k) f_j(x_k), \tag{9}$$

$$\frac{\partial^2 C}{\partial w_i \partial w_j} = \frac{1}{K} \sum_k p_k (1 - p_k) f_i(x_k) f_j(x_k) \tag{10}$$

Denote $\mathbf{w} = [w_1, \cdots, w_J]^T$ as the parameter vector; denote $\mathbf{p} = [p_1, \cdots, p_K]^T$ and $\mathbf{t} = [t_1, \cdots, t_K]^T$ as the likelihood and label vector of all examples; and denote $\mathbf{F}$ as the $K \times J$ design matrix with $f_j(x_k)$ as the $(k, j)^{th}$ element. We have in vector form:

$$\nabla C(\mathbf{w}) = \frac{1}{K} \mathbf{F}^T (\mathbf{p} - \mathbf{t})$$
$$\mathbf{H}_C(\mathbf{w}) = \nabla \nabla C(\mathbf{w}) = \frac{1}{K} \mathbf{F}^T \mathbf{R} \mathbf{F}, \tag{11}$$

where $\mathbf{R}$ is the $K \times K$ diagonal *weighting* matrix with elements $R_{kk} = p_k(1 - p_k)$.

It is therefore straightforward to apply Eq. (3) for logistic regression adaptation. Following the idea in Section 2.3, the cost function on the old data-set can be approximated as:

$$C(\mathbf{w}) \approx C(\mathbf{w}^{(o)}) + \nabla C(\mathbf{w}^{(o)})(\mathbf{w} - \mathbf{w}^{(o)}) + \frac{1}{2}(\mathbf{w} - \mathbf{w}^{(o)})^T \mathbf{H}_C(\mathbf{w}^{(o)})(\mathbf{w} - \mathbf{w}^{(o)}), \tag{12}$$

where $\nabla C(\mathbf{w}^{(o)})$ and $\mathbf{H}_C(\mathbf{w}^{(o)})$ are computed at the generic classifier's weight vector $\mathbf{w}^{(o)}$ on the old data-set. The cost function on the new data-set depends on the form of user labels. Next, we present two types of labels that can be used in our adaptation framework: direct labels and similarity labels. Note both types of labels can be obtained automatically, via co-training [18, 21] or tracking (Section 5.2).

3.2.1 Direct Labels

Direct labels are labels given on examples directly, *e.g.*, $\mathcal{S}^{(n)} = \{(x_k^{(n)}, t_k^{(n)}), k = 1, \cdots, K^{(n)}\}$, where $x_k^{(n)}$ is the example and $t_k^{(n)}$ is the label information. Such labels are identical to those used for training the generic detector before adaptation. The cost function on the new data-set can be the same cross entropy error function defined in Eq. (8):

$$D \triangleq -\frac{1}{K^{(n)}} \sum_k \{t_k^{(n)} \ln p_k^{(n)} + (1 - t_k^{(n)}) \ln(1 - p_k^{(n)})\}, \tag{13}$$

where $p_k^{(n)}$ is defined as in Eq. (6). The overall cost function for classifier adaptation is hence:

$$J(\mathbf{w}) = C(\mathbf{w}^{(o)}) + \nabla C(\mathbf{w}^{(o)})(\mathbf{w} - \mathbf{w}^{(o)}) + \frac{1}{2}(\mathbf{w} - \mathbf{w}^{(o)})^T \mathbf{H}_C(\mathbf{w}^{(o)})(\mathbf{w} - \mathbf{w}^{(o)}) + \lambda D(\mathbf{w}). \tag{14}$$

We minimize the overall cost function by an efficient iterative technique based on the *Newton-Raphson* iterative optimization scheme. It takes the form:

$$\mathbf{w}^{[i+1]} = \mathbf{w}^{[i]} - \mathbf{H}_J^{-1}(\mathbf{w}^{[i]})\nabla J(\mathbf{w}^{[i]}), \tag{15}$$

where i is the iteration index. It is easy to compute:

$$\begin{aligned} \nabla J(\mathbf{w}^{[i]}) &= \mathbf{H}_C(\mathbf{w}^{(o)})(\mathbf{w}^{[i]} - \mathbf{w}^{(o)}) + \\ &\quad \nabla C(\mathbf{w}^{(o)}) + \lambda \nabla D(\mathbf{w}^{[i]}), \\ \mathbf{H}_J(\mathbf{w}^{[i]}) &= \mathbf{H}_C(\mathbf{w}^{(o)}) + \lambda \mathbf{H}_D(\mathbf{w}^{[i]}), \end{aligned} \tag{16}$$

where the gradient and Hessian of the error function $D(\mathbf{w})$ *on the new data-set* can be computed as in Eq. (11).

During the iterative optimization process, the weight vector of the generic classifier is used for initialization:

$$\mathbf{w}^{[0]} = \mathbf{w}^{(o)}. \tag{17}$$

We iterate on Eq. (15) until the Newton decrement is less than a certain threshold ξ:

$$\sqrt{\nabla J(\mathbf{w}^{[i]})^T \mathbf{H}_J^{-1}(\mathbf{w}^{[i]}) \nabla J(\mathbf{w}^{[i]})} < \xi. \tag{18}$$

3.2.2 Similarity Labels

Instead of labeling examples with positive or negative tags, one may also specify similarity labels, which indicates whether two examples share the same direct label or not. Similarity label takes the form of $\mathcal{S}^{(n)} = \{(x_{k1}^{(n)}, x_{k2}^{(n)}, z_k^{(n)}), k = 1, \cdots, K^{(n)}\}$, where $x_{k1}^{(n)}$ and $x_{k2}^{(n)}$ are the two examples, $z_k^{(n)} = 1$ indicates the two examples should have the same label, and $z_k^{(n)} = 0$ indicates the two examples should have different labels. In the following, we skip the superscript $^{(n)}$ to simplify the notations.

The probability of x_{k1} and x_{k2} sharing the same label can be written as:

$$p_k = p_{k1}p_{k2} + (1 - p_{k1})(1 - p_{k2}), \tag{19}$$

where

$$p_{kl} = \frac{1}{1 + \exp\{-\sum_j w_j f_j(x_{kl})\}}, l \in 1, 2. \tag{20}$$

The cross-entropy error function is again:

$$D \triangleq -\frac{1}{K} \sum_k \{z_k \ln p_k + (1 - z_k) \ln(1 - p_k)\}. \tag{21}$$

We still resort to the Newton-Raphson method to find the optimal parameter vector as in Eq. (15) and (16), except that the gradient and Hessian of the cost function on the new data-set needs to be revised. The gradient of the cost function on the new data-set is:

$$\frac{\partial D}{\partial w_j} = \frac{1}{K} \sum_k \frac{(p_k - z_k) g_{jk}}{r_k}, \tag{22}$$

where

$$\begin{aligned} r_k &= p_k(1 - p_k), \\ g_{jk} &= \frac{\partial p_k}{\partial w_j} = -q_{k2} r_{k1} f_j(x_{k1}) - q_{k1} r_{k2} f_j(x_{k2}), \end{aligned}$$

with $r_{kl} = p_{kl}(1 - p_{kl})$, $q_{kl} = \frac{\partial r_{kl}}{\partial p_{kl}} = 1 - 2p_{kl}, l \in \{1, 2\}$. Second order derivatives are:

$$\frac{\partial^2 D}{\partial w_i \partial w_j} = \frac{1}{K} \sum_k \left\{ \frac{[p_k^2 + z_k q_k] g_{jk} g_{ik}}{r_k^2} + \frac{(p_k - z_k) h_{ijk}}{r_k} \right\}, \tag{23}$$

where

$$\begin{aligned} q_k &= \frac{\partial r_k}{\partial p_k} = 1 - 2p_k, \\ h_{ijk} &= \frac{\partial g_{jk}}{\partial w_i} \\ &= 2 r_{k1} r_{k2} [f_i(x_{k1}) f_j(x_{k2}) + f_i(x_{k2}) f_j(x_{k1})] - \\ &\quad q_{k1} q_{k2} [r_{k1} f_i(x_{k1}) f_j(x_{k1}) + r_{k2} f_i(x_{k2}) f_j(x_{k2})] \end{aligned}$$

Note the Hessian matrix for similarity labels on the new data-set is not necessarily positive definite, hence optimizing the error function $D(\mathbf{w})$ on the new data-set alone does not ensure a global minimum. Fortunately, $\mathbf{w}^{(o)}$ is the global optimal estimate minimizing the error function on the old data-set and serves well as a good initial estimate in the optimization for adaptation.

3.3. Boosting Classifier and Its Adaptation

In a typical boosting classifier, each example is classified by a linear combination of weak classifiers. Given a test example x_k, define the *score* of the example s_k as a weighted sum of weak classifiers $h_j(\cdot)$, *i.e.*,

$$s_k = \sum_j \alpha_j h_j(x_k) \tag{24}$$

where $h_j(x_k)$ can be written as:

$$h_j(x_k) = \begin{cases} +1 & \text{if } h_j(x_k) > t_j \\ -1 & \text{otherwise} \end{cases} \tag{25}$$

where t_j is the threshold for weak classifier $h_j(\cdot)$. The final decision is made by comparing the example's score with an overall threshold T. That is, if $s_k > T$, then example x_k is a positive example; otherwise, x_k is a negative example.

There have been many approaches proposed in literature on how to effectively learn a boosting classifier, *e.g.*, AdaBoost [8], GentleBoost [9], MILBoost [23], BrownBoost [7], *etc.* Friedman *et al.* showed in [9] that the AdaBoost algorithms are indeed Newton methods for optimizing a particular exponential loss function – a criterion which behaves much like the log-likelihood on the logistic scale.

In [16], Mason *et al.* showed that boosting can be viewed as a gradient-decent algorithm in the function space. We follow Mason's AnyBoost framework and define the probability of an example being positive as:

$$p_k = \frac{1}{1 + \exp\{-s_k\}}, \tag{26}$$

Subsequently, we can use gradient-decent to search for the weak classifiers $h_j(\cdot)$ and the weights α_j with the same cost function as in Eq. (8) [16].

Under the above AnyBoost formulation, if we restrict ourselves to updating only the weights α_j, the adaptation of a boosting classifier can be identical to the adaptation of the logistic regression classifier. The difference is that in logistic regression, the features $f_j(\cdot)$ are usually real-valued. In contrast, in a boosting classifier, the weak classifiers $h_j(\cdot)$ are binary. However, this has no impact on the application of adaptation on boosting classifiers.

In some recent work for tracking [11, 15], the weak classifiers were also updated during tracking. One approach is to compute the gradient of the cost function over weak classifiers numerically, as was done in [15]. Alternatively, one can record the compact information (*e.g.*, the gradient and the Hessian) for all possible combinations of weak classifiers. This may result in a large amount of information to be sent to the end user, hence its application for classifier adaptation may be limited.

4. Related Work

Classifier adaptation, as presented in the form of Section 2.2, can be viewed as a very general formulation of classifier learning. For instance, if we set the parameter λ in Eq. (3) very large, minimizing Eq. (3) is equivalent to training a classifier on the new data-set only. This may indeed be the best practice if sufficient amount of data are collected in the test environment. On the other hand, if the new data-set contains examples given one-by-one, and the adaptation algorithm runs for every new input example, the methodology shown in Section 2.2 can be used to explain many online learning or sequential learning algorithms.

Many recent approaches that involve online learning for detection/tracking [4, 1, 11, 14, 15, 20] are related to our work. However, we found some of these methods (*e.g.*, [1, 15]) do not consider the cost function on the old data-set. Instead, they take the feature set from the previous frame, update it if necessary, and then learn a new weight vector solely based on the tracked result in the current frame. This approach can cause model drifting in tracking. Collins and Liu [4] addressed this issue by using the first frame as an "anchor" frame, effectively always keeping a small old data-set in its original form. In [14], the generic person detector is in fact never updated. The authors relied on some simple classifiers (*e.g.*, LDA) to learn the appearance change of the tracked object, and combine it with the generic person detector to avoid drifting. While this approach is interesting, we consider it a very special case of classifier adaptation that operates on the *fused* classifier instead of the generic person detector.

Another family of related work is co-training based detector adaptation or improvement [13, 18, 21, 12, 6], or more generally, semi-supervised learning [24]. In co-training, multiple independent classifiers are applied on the same examples. If some of the classifiers have high confidence on a particular example, it can be used to update the remaining classifiers. Therefore, co-training is a mechanism to obtain additional training examples from unlabeled data, and our proposed adaptation algorithm can be applied once these examples are obtained. As for the adaptation algorithms used after new examples are obtained, [18] used an online Winnow algorithm to update the classifier; [12, 21] used online boosting based on the work in [19]. Our proposed Taylor expansion based classifier adaptation algorithm can be applied to both, and many other classifiers such as linear discriminant analysis, neural networks, logistic regression, *etc*. In addition, our algorithm can handle data with similarity labels, which cannot be used for traditional online Winnow or boosting.

Interestingly, the Taylor expansion based adaptation can also be viewed as a regularization method for parametric learning. Take logistic regression as an example. Using the Bayesian formulation, it can be shown that a Gaussian or Laplacian prior of the parameter vector can lead to a logistic regression formulation with L2 or L1 regularization [2]. Our proposed method can be viewed as using the Hessian matrix on the old data-set to regularize the logistic regression optimization on the new data-set. It certainly belongs to the category of L2 regularization. However, unlike the widely used i.i.d. Gaussian prior [17], the Hessian matrix from the old data-set contains more information about each parameter and their correlations, hence it can serve better when the new data-set is small.

5. Experimental Results

We test the proposed classifier adaptation algorithm on the task of detecting people from panoramic videos of conference rooms, as shown in Figure 1. The videos are captured at resolution 1056×144. The task is very challenging due to pose variations, occlusions, small head sizes (*e.g.*, as small as 10×10 pixels), non-static background (*e.g.*, moving chairs and monitor contents), lighting variation, *etc*.

A generic boosting-based person detector is trained on 93 meetings (6350 labeled frames) collected in more than 10 different rooms. In every labeled frame each person is marked by a hand-drawn box around the head of the person. The blue box in Figure 1 shows such an example. Since the head size can be very small, we expand the ground truth

Figure 1. Example panoramic views of two meeting rooms for person detection. In the upper image, the blue box is one of the hand-labeled ground truth heads. The red box is the expanded box that includes shoulder for detector training.

box to include people's shoulders, making it effectively an upper body detector (see the red box in Figure 1). The minimum detection window size is 35×35, and the 93 sequences comprise over 100 million examples (both positive and negative) for training.

In addition to the monochrome images, two additional feature images are used. One measures the difference between two subsequent video frames and the other measures the long term average of the temporal difference frames. A set of 6946 Haar like features are extracted from these three images and serve as the feature pool for boosting. We adopt the logistic variation of Adaboost developed by Collins, Schapire and Singer [3] to train the classifier. The resultant detector has a total of 120 weak classifiers.

The generic person detector performs well in unseen meeting rooms. However, since a meeting room often has limited variations in background and lighting, we believe a classifier adaptation algorithm can help improve the generic detector's performance further. We collected a total of 17 meetings in the same meeting room (upper image of Figure 1) during a period of one week. A total of 265 frames are labeled as before. The 17 meetings are randomly split into two groups: 8 meetings (128 labeled frames) for adaptation and 9 meetings (137 labeled frames which contain 527 persons in total) for testing. The adaptation is performed on the weights (α_j in Eq. (24)) of the weak classifiers only, and the optimization is done through the Newton-Raphson scheme (Eq. (15) and (16)).

5.1. Results on Direct Labels

In the first set of experiments we assume the end user has provided direct labels for some of the frames in the adaptation data-set. Figure 2 shows the adapted detectors' receiver operating characteristic (ROC) curves on the 9 testing sequences for $\lambda = 0.5$ and 5.0. In each figure, we gradually increase the number of labeled frames in order to observe the impact of the amount of labeled data on the adaptation performance. Note each labeled frame contributes around 16,000 labeled examples for training. When all 128 frames are used for adaptation, the total number of labeled examples is over 2 million. For the curves that use N (where $N = 2, 4, 8, 16, 32$) labeled frames, we randomly sample N labeled frames from the adaptation data-set to perform classifier adaptation. Each ROC curve in Figure 2 is the *average* of 100 trials of such random sampling.

N	2	4	8	16	32	all
λ	$\leq$0.1	$\leq$0.1	0.1-0.5	0.5-1.0	0.5-1.0	5.0-10.0

Table 1. Optimal λ range for different amount of ground truth data. N is the number of labeled frames used for adaptation.

From Figure 2, it can be seen that in general the more labeled frames one has the better the performance is after adaptation. When $\lambda = 0.5$, with 4 newly labeled frames in the test environment the adapted classifier already outperforms the generic detector. When all frames in the adaptation data-set are used, we observe the detection accuracy improves from 85.6% to 90.4% with 30 false detections ($\lambda = 5.0$), a decrease of around 33.3% in detection error.

Figure 3 shows two sets of curves when the number of labeled frames are 8 and 128 (all available frames). We vary the parameter λ from 0.1 to infinity. As mentioned earlier, when λ is infinite classifier adaptation is equivalent to re-training a detector on the newly labeled data only. It can be seen that in both sets of curves, setting λ to infinity does not achieve the best performance. This demonstrates the necessity of having the Hessian of the generic classifier as the regularization term. In Table 1, we show the optimal λ range with respect to the number of labeled frames for adaptation. It can be seen that when the number of labeled frames is small, a small λ tends to give better results. On the other hand, if a large number of labeled frames are available, a large λ tends to perform better.

5.2. Results on Similarity Labels

We next report our experimental results on similarity labels. All 128 frames from the 8 adaptation sequences are used in the following experiments. These frames are organized into 120 pairs, where each pair consists of two subsequent frames in the *same* meeting sequence. Note, however, that a pair of subsequent frames may be a few seconds apart.

In the first experiment, we give ground truth similarity labels to the adaptation algorithm. For each pair of frames, we choose a detection window from one frame, and randomly choose another detection window from the other frame. The similarity label is computed from the actual direct labels of the two windows, *i.e.*, 1 if they have the same direct label and 0 otherwise. This sampling process continues until all detection windows in the first frame have been selected. In total around 2 million example pairs are created for training (most of them are pairs of negative examples).

The data collected above are then fed into the classifier adaptation algorithm as described in Section 3.2.2. Figure 4 shows the detector performance after adaptation. Note with all λ values, the adapted detector outperforms the generic detector (Figure 4(a)). Figure 4(b) gathers the best adapted

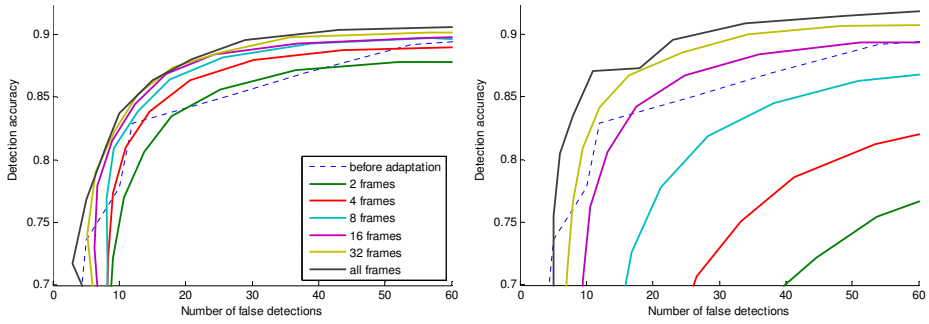

Figure 2. Adapted person detector performance with direct labels. In each figure λ is fixed. The two figures share the same legend.

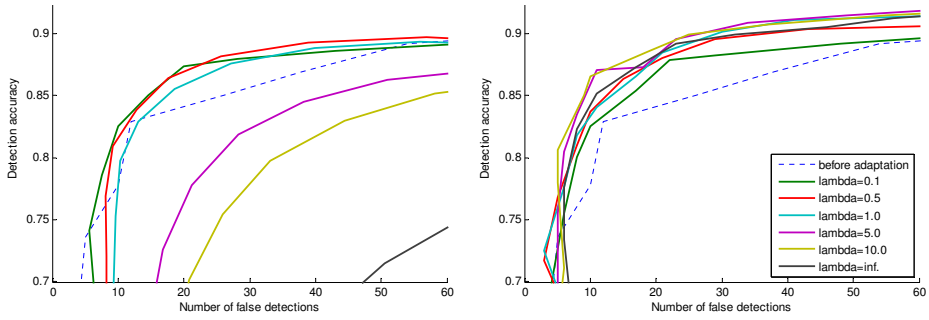

Figure 3. Adapted person detector performance with direct labels. In each figure the number of labeled frames is fixed. The two figures share the same legend.

detector with direct labels ($\lambda = 10.0$) and similarity labels ($\lambda = 5.0$). The two ROC curves are very similar, hence similarity labeling is *as effective as* direct labeling when used for adaptation.

Finally, we present a simple algorithm to generate similarity labels automatically for adaptation. First, define the *histogram distance* between two detection windows as follows. Let the histogram of the first window be $\mathbf{p} = \{p(u), u = 1, \cdots, m\}$, where m is the total number of bins, and the histogram of the second window be $\mathbf{q} = \{q(u), u = 1, \cdots, m\}$. The distance between the two discrete distributions is defined as [5]:

$$d = \sqrt{1 - \rho[\mathbf{p}, \mathbf{q}]}, \tag{27}$$

where

$$\rho[\mathbf{p}, \mathbf{q}] = \sum_{u=1}^{m} \sqrt{p(u)q(u)} \tag{28}$$

is the sample estimate of the Bhattacharyya coefficient between $\mathbf{p}$ and $\mathbf{q}$.

For each pair of frames, we choose a detection window from one frame, and search for a window (with identical size) in the other frame that has the smallest histogram distance to the first window. Once the window with the smallest distance has been found, we put the example pair in the training set with label 1 if the distance $d < 0.03$. Over 1 million example pairs are constructed automatically, among which 10 pairs have the wrong labels ($\sim 0.001\%$).

Figure 5 shows the adapted detector's performance when varying λ. Note the adapted detector outperforms the generic detector when $\lambda < 0.1$. The small λ values indicate that we shall rely more on the generic detector when the similarity labels are provided automatically by a rudimentary algorithm such as this one. We expect a more elaborated algorithm for generating the similarity labels may lead to further improvements in the detector performance.

6. Conclusions and Future Work

We have presented a general framework for classifier adaptation, and a Taylor expansion based solution that is applicable to various classifiers and user labels. The idea is to combine the cost function on the old and new training data-set, and approximate the cost function on the old data-set with a Taylor expansion representation. In this way, only the gradient and Hessian of the generic classifier need to be sent to the end user for adaptation. The algorithm is general enough to handle different classifiers such as logistic regression and boosting, and different user labels such as direct labels and similarity labels.

A few challenges remain as our future work. For instance, while the rule of thumb is to use a small λ value when the amount of data collected in the new test envi-

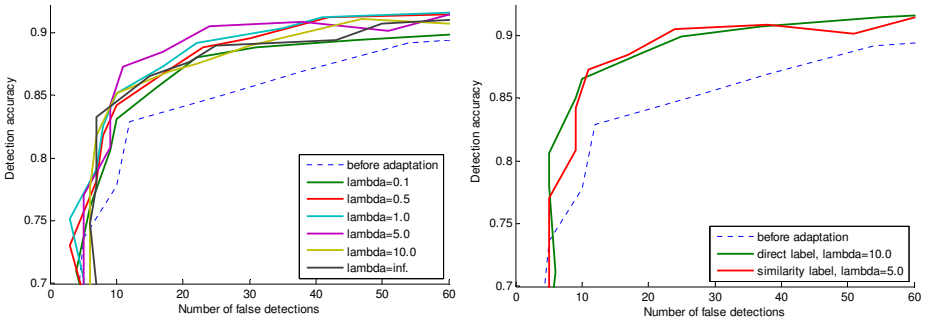

Figure 4. Adapted person detector performance with ground truth similarity labels. (a) Using all frames for adaptation, the performance of the adapted classifier with respect to different λ. (b) Best adaptation performance comparison using direct labels and similarity labels.

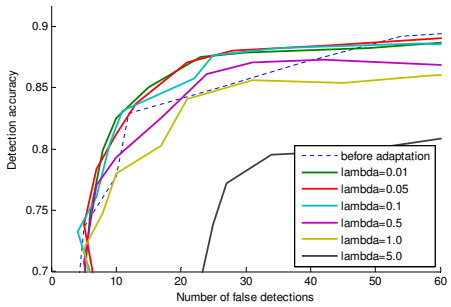

Figure 5. Adapted person detector performance with automatically generated similarity labels.

ronment is small (as shown in Table 1 based on our empirical study), determining the optimal λ is still a difficult task. Also, the adapted classifier may require a different final threshold. Typically the threshold is derived from a validation data-set, but such a data-set may be difficult to obtain without significant user effort.

References

[1] S. Avidan. Ensemble tracking. In *CVPR*, 2005. 1, 5

[2] C. M. Bishop. *Pattern Recognition and Machine Learning*. Springer Science+Business Media, LLC, 2006. 3, 5

[3] M. Collins, R. Schapire, and Y. Singer. Logistic regression, adaboost and bregman distances. *Machine Learning*, 48(1-3):253–285, 2002. 6

[4] R. T. Collins and Y. Liu. Online selection of discriminative tracking features. In *ICCV*, 2003. 1, 5

[5] D. Comaniciu, V. Ramesh, and P. Meer. Kernel-based object tracking. *IEEE Trans. on PAMI*, 25(5):564–577, 2003. 7

[6] C. O. Conaire, N. E. O'Connor, and A. F. Smeaton. Detector adaptation by maximising agreement between independent data sources. In *CVPR*, 2007. 5

[7] Y. Freund. An adaptive version of the boost by majority algorithm. In *Proc. of the 12th Conf. on Computational Learning Theory*, 1999. 4

[8] Y. Freund and R. E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139, 1997. 4

[9] J. Friedman, T. Hastle, and R. Tibshirani. Additive logistic regression: A statistical view of boosting. Technical report, Dept. of Statistics, Stanford University, 1998. 4

[10] P. E. Gill, W. Murray, and M. H. Wright. *Practical Optimization*. Academic Press, 1981. 3

[11] H. Grabner and H. Bischof. On-line boosting and vision. In *CVPR*, 2006. 1, 5

[12] O. Javed, S. Ali, and M. Shah. Online detection and classification of moving objects using progressively improving detectors. In *CVPR*, 2005. 2, 5

[13] A. Levin, P. Viola, and Y. Freund. Unsupervised improvement of visual detectors using co-training. In *ICCV*, 2003. 2, 5

[14] Y. Li, H. Ai, T. Yamashita, S. Lao, and M. Kamade. Tracking in low frame rate video: a cascade particle filter with discriminative observers of different life spans. In *CVPR*, 2007. 1, 5

[15] X. Liu and T. Yu. Gradient feature selection for online boosting. In *ICCV*, 2007. 1, 5

[16] L. Mason, J. Baxter, P. Bartlett, and M. Frean. Boosting algorithms as gradient decent. In *NIPS*, 2000. 5

[17] T. P. Minka. Algorithms for maximum-likelihood logistic regression. Technical report, Dept. of Statistics, Carnegie Mellon University, 2001. 3, 5

[18] V. Nair and J. J. Clark. An unsupervised, online learning framework for moving object detection. In *CVPR*, 2004. 2, 3, 5

[19] N. C. Oza. *Online Ensemble Learning*. PhD thesis, University of Calfornia, Berkeley, 2002. 5

[20] M.-T. Pham and T.-J. Cham. Online learning asymmetric boosted classifiers for object detection. In *CVPR*, 2007. 5

[21] P. M. Roth, H. Grabner, D. Skočaj, H. Bischof, and A. Leonardis. Online conservative learning for person detection. In *Proc. 2nd Joint IEEE International Workshop on VS-PETS*, 2005. 3, 5

[22] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *CVPR*, 2001. 1

[23] P. Viola, J. C. Platt, and C. Zhang. Multiple instance boosting for object detection. In *NIPS*, 2005. 4

[24] X. Zhu. Semi-supervised learning literature survey. Technical report, 1530, Dept. of Computer Science, University of Wisconsin - Madison, 2007. 5