Unsupervised and Online Update of Boosted Temporal Models: the *UAL$_2$Boost*

Pedro Canotilho Ribeiro, Plinio Moreno, José Santos-Victor
Instituto Superior Técnico
Instituto de Sistemas e Robótica
Lisboa, Portugal
{*pribeiro, plinio, jasv*}*@isr.ist.utl.pt*

***Abstract*—The application of learning-based vision techniques to real scenarios usually requires a tunning procedure, which involves the acquisition and labeling of new data and in situ experiments in order to adapt the learning algorithm to each scenario.**

We address an automatic update procedure of the L$_2$boost algorithm that is able to adapt the initial models learned off-line. Our method is named *UAL$_2$Boost* and present three new contributions: (i) an on-line and continuous procedure that updates recursively the current classifier, reducing the storage constraints, (ii) a probabilistic unsupervised update that eliminates the necessity of labeled data in order to adapt the classifier and (iii) a multi-class adaptation method.

We show the applicability of the on-line unsupervised adaptation to human action recognition and demonstrate that the system is able to automatically update the parameters of the L$_2$boost with linear temporal models, thus improving the output of the models learned off-line on new video sequences, in a recursive and continuous way. The automatic adaptation of *UAL$_2$Boost* follows the idea of adapting the classifier incrementally: *from simple to complex*.

***Keywords*-online, unsupervised and semi-supervised learning; L2 boosting; multi-class human action classification**

I. Introduction

Cognitive and evolutionary processes allow living beings to adapt models acquired by the brain in a continuous way. These adaptation processes allows biological systems to maintain classification and detection performances under large modifications on the environment, and consequently enables a high level of generalization. Recent research on semi-supervised learning aims to provide these generalization capabilities to machines, by considering both labeled and unlabeled data to improve the performance of standard machine learning algorithms [1], [2], [3]. Most of the semi-supervised approaches belong to the class of offline learning problems, where all the available data (labeled and unlabeled) is fixed at the beginning [1], [2], [4]. However, this is an unrealistic assumption for video-based scenarios, where online classifiers fit better due to their intrinsic update capability [5], [6], [7]. An application oriented scenario for semi-supervised learning should follow the divide and conquer approach using two phases: (a) the offline learning using labeled data and (b) the online and continuous adaptation using the unlabeled data.

There are recent works that adopt the same idea [8], [9], nevertheless they are tailored to a problem-specific binary detection task, which needs the tunning of several parameters.

In this paper we propose an application of multi-class semi-supervised learning, incorporating an automatic tunning of classifiers during the online phase. The goal of this application is to reduce the in situ experiments, which require acquisition, labeling and (re)training of the classifiers on the new scenarios. For instance, it is very difficult to consider all the possible variations in the visual appearance of an object due to illumination conditions (appearance, speed, etc.). Instead, if we consider the most representative changes and let the semi-supervised tunning adapt the object model to a particular scenario, the requirements of in situ experiments would be ideally none.

We address the general problem of online unsupervised adaptation of the base learners of the L$_2$ boosting algorithm [10] and apply the algorithm on the unsupervised adaptation of models for human action recognition, from video, and evaluate the algorithm in two scenarios: (i) an adaptable scenario, and (ii) a non-adaptable scenario.

A. Related work

The state-of-the-art semi-supervised boosting methods such as SemiBoost [1] and ASSEMBLE [2] work in a batch fashion, so these models do not take advantage of new data. ASSEMBLE aims to improve the classification margin of boosting by selecting unlabeled examples with large classification confidence and assigning them the class label (pseudolabel), which are provided by the current classifier. The labeled data along with the selected pseudolabeled data are utilized in the next iteration to train a second classifier. The process is repeated for the other boosting iterations. SemiBoost defines a confidence measure that takes into account two sources: the strong classifier output and the similarity matrix, and the pseudolabel is computed from this confidence measure. Furthermore, SemiBoost is designed to wrap around any given supervised algorithm and adapt it to use also unsupervised data. An application of Semiboost for visual detection [4] demonstrate its improvement capabilities, thus we present a direct comparison of this state-of-the-art batch method against our *UAL$_2$Boost*.

Javed et al. [7], use the online boosting method proposed by Oza ([5]) in conjunction with the co-training idea, introducing a method that adapts previous boosting classifiers online and in an unsupervised manner for binary problems. Their basic idea is to use some weak learners to co-train all the others for the points that introduce new information to the problem, i.e. that are confidently classified by some weak learners but not by the strong learner (the final classifier), guarantying that the adapted points are near the decision boundary. They propose to use a validation set that defines the confidence thresholds, which in the end adapts a small amount of the observed data points. In addition, the use of a fixed validation set to define the adaptation process is biasing the adapted data to be, somehow, similar to that set.

The adaptation of offline learned prototypes for specific binary problems was proposed by [8], [9]. Wu and Nevatia [9] proposed the development of an oracle with high precision that is able to adapt the simple features and the complexity of the classifiers. Although the system is able to: (i) perform online learning on a cascade-structured detector and (ii) integrate noise restraining strategies, it is not applicable to general machine learning problems. Their application is restricted binary problems with cascade structures and involves the tunning of a large number of parameters. Furthermore, and similarly to the method proposed by Huang et al. [8], the selection of the samples for adaptation does not consider a probability-based measure, they just pick the *best* online samples to adapt. Thus, the adaptation procedure needs to be tuned for each particular problem.

B. Our approach

The semi-supervised boosting algorithms [2], [1] are based on the idea of, at round m, compute the current strong classifier output to: (i) predict the class of unlabeled data and (ii) obtain a margin-based confidence weight. Then, the unlabeled data with their predicted labels (pseudolabels) and weights are combined with the labeled data in order to select the weak learner at round $m+1$. Similar to previous works, we build our unsupervised adaptation using the pseudolabels and confidence weights. In difference to previous approaches, we do not add new weak learners to adapt the new data, but we: (a) update the current weak learners and (b) propose a margin-based confidence weight for multi-class problems. We propose an algorithm based on the L_2Boost learning, due to the simpler weighting process in comparison to AdaBoost, and compensate for the use of an quadratic loss function for classification using a $[-1$ $1]$ constrained classifier at each round. Our proposal is an Unsupervised Adaptation of L_2Boost weak learners, the *UAL$_2$Boost* algorithm, for binary and one vs. all multi-class problems. Figure 1 depicts the two steps of our approach: (i) the offline learning and (ii) the unsupervised adaptation. The first step consists of the offline supervised learning of a classifier *prototype*, which has two types of parameters for each iteration: The feature dimension j and its corresponding model β (Θ_j, and Θ_β denote the parameters of all the iterations). This step is basically supervised learning to obtain the *prototype* using the labeled database, (I, Y). The second step is an unsupervised online process (see the bottom row of Fig. 1), which incorporates the ongoing data samples into the previously learned classifier by adapting the models, Θ_β. These new data points do not have label, so the adaptation of the model Θ_β uses the *pseudo-label* (Y) provided by the previous classifier, weighted by the margin-based confidence measure Γ. Then, the adapted models Θ_β become the current classifier when the next data sample arrives.

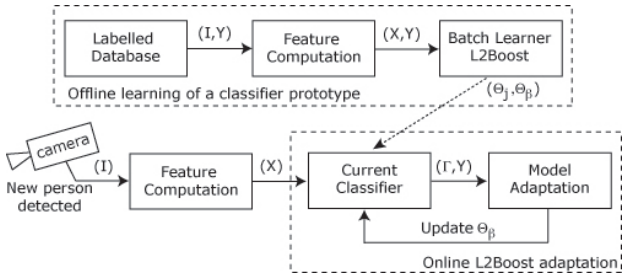


Figure 1. Global description of the learning framework. The top row describes the supervised learning of a classifier *prototype* and the bottom row the ongoing update of the models using all the observed data.

II. Supervised L_2Boost with temporal models

In a formal description, the L2boost algorithm, for binary problems, estimates the function $F : \mathbb{R}^d \rightarrow \mathbb{R}$ by minimizing the expected cost $\mathbb{E}[C(y, F(X))]$ based on the data $(y_i, X_i), i = 1, ..., n$. The cost function is $C(y, f) = (y - f)^2/2$ with $y \in \{-1, 1\}$ and its respective population minimizer is $F(X) = \mathbb{E}[y|X = x]$. The overall optimization is achieved by means of a sequential stagewise approximation along M rounds, optimizing a so called weak learner in each round, m [10].

We use a featurewise implementation that optimizes each weak learners to be a linear temporal model for the feature that achieves less cost: $f_m(X_i) = X_i^{j^m}\beta^m$, with $X_i^{j^m} \in \mathbb{R}^{T+1}$, $\beta^m \in \mathbb{R}^{T+1}$ and j^m the feature chosen in round m from the set of D features. In order to use matrix notation, we stack all the y_i values into the vector $Y \in \mathbb{R}^N$ and all the X_i data points into the matrix $X \in \mathbb{R}^{N \times D \times T+1}$ (see chapter 5 for a detailed explanation of the feature computation and structure). This means that, at each round m, we optimize a temporal model β for each possible feature $j = 1, ..., D$, choosing the one that achieve less error:

$$\widehat{\beta} = \arg\min_{\beta,j}(Y - X^j\beta)^T(Y - X^j\beta). \quad (1)$$

The solution is $\widehat{\beta}^m = (X^{j^m T}X^{j^m})^{-1}X^{j^m T}Y$, where j^m is the feature that achieves less error, for a specific round m, and $\widehat{\beta}^m$ the corresponding temporal model for that feature.

The featurewise L_2boosting algorithm with linear temporal models is as follows:

1) **Initialization** Chose M and set m=0. Given data (Y, X), fit the first weak learner, $\widehat{F}_0 = X^{j^0}\widehat{\beta}^0$. β^0 and j^0 are computed from Eq. 1.
2) **Projection of gradient to learner** Compute the negative gradient (in this case are the residuals) $u_i^{m+1} = y_i - \widehat{F}_m(X_i)(i = 1, ..., n)$. For simplicity, stack all u_i values into the vector $U \in \mathbb{R}^N$.
   Use the residuals U^{m+1} to fit the learner $\widehat{f}_{m+1} = X^{j^{m+1}}\widehat{\beta}^{m+1}$ changing Y for U in Eq. 1.
   Update $\tilde{F}_{m+1} = \widehat{F}_m + \widehat{f}_{m+1}$. Make $\widehat{F}_{m+1} = sign(\tilde{F}_{m+1})\min(1, |\tilde{F}_{m+1}|)$.
3) **Iteration** If $m + 1 < M$ increase m by 1 and goto step2. If $m + 1 = M$ return $\Theta_j = \{j^0, ..., j^m, ...\}$, and one set of models, $\Theta_\beta = \{\beta^0, ..., \beta^m, ...\}$

The classification of a new point X_i is given by the sign of the strong classifier result, $\operatorname{sgn}\widehat{F}_M(X_i)$, but notice that at each round the classifier is constraint to be in $[-1\ 1]$. This procedure works better when using L_2Boost for classification because it compensates for the use of a quadratic loss function, and it's called L_2Boost with constraints [10]. The strong classifier $F(x)$ relates the class-conditional probabilities,

$$F(x) = 2p(y = 1|x) - 1, |F(x)| = |p(y = 1|x) - p(y = -1|x)|, \quad (2)$$

and its module $|F(X_i)|$ is the classification margin, that is the probability of labeling the new data point given the models estimated.

III. Unsupervised Adaptation of L_2Boost: the *UAL2Boost* algorithm

A. Online update of learned models

For now lets assume that the data comes with labels, but the next section relaxes this assumption allowing to adapt observed data without supervision. The objective here is to update the set of temporal models Θ_β, using the previously learned set of features, Θ_j. The update is performed without storing any of the previous data points. For that, let us parameterize β^m as

$$\beta^m = P^{m-1}s^m = (X^{j^m T}X^{j^m})^{-1}X^{j^m T}U^m,$$

where $P^m = X^{j^m T}X^{j^m}$ and $s^m = X^{j^m T}U^m$, which allows to express β^m in recursive summations. Let us assume that we have seen data X_τ so far, and new data $X_{\tau+1}$ just arrived. Thus, the complete dataset matrix is $X = \left[X_\tau^T X_{\tau+1}^T\right]^T$. In order to update the parameters, Θ_β, we must minimize the residuals, $U = \left[U_\tau^T U_{\tau+1}^T\right]^T$. The updated parameters are

$$P^m = P_\tau^m + P_{\tau+1}^m, \quad s^m = s_\tau^m + s_{\tau+1}^m, \quad (3)$$

where $P_\tau^m = X_\tau^{j^m T}X_\tau^{j^m}$, $P_{\tau+1}^m = X_{\tau+1}^{j^m\ T}X_{\tau+1}^{j^m}$, $s_\tau^m = X_\tau^{j^m T}U_\tau^m$ and $s_{\tau+1}^m = X_{\tau+1}^{j^m\ T}U_{\tau+1}^m$. Considering the update procedure of Eq. (3), the online integration of new labeled data is as follows:

1) **Batch L2 boost** Given an initial training set $\{(Y_\tau, X_\tau)\}$, compute the parameters $\{P_\tau^m, s_\tau^m, \Theta_j, m = 0, \ldots, M-1\}$ by applying the algorithm of Section II.
2) **Initialization** Set $m = 0$. Wait for new data $(X_{\tau+1}, Y_{\tau+1})$ to arrive, then fit the first weak learner $\widehat{F}_0 = X^{j^0}\widehat{\beta}^0$ by updating the model $\widehat{\beta}^0$.
   $P_{\tau+1}^0 = X_{\tau+1}^{j^0\ T}X_{\tau+1}^{j^0}$, $s_{\tau+1}^0 = X_{\tau+1}^{j^0\ T}Y_{\tau+1}$, $\widehat{\beta}^0 = (P_\tau^0 + P_{\tau+1}^0)^{-1}(s_\tau^0 + s_{\tau+1}^0)$.
3) **Adaptation of the linear models**
   $U_{\tau+1}^{m+1} = Y_{\tau+1} - \hat{F}_m(X_{\tau+1})$,
   $P_{\tau+1}^{m+1} = X_{\tau+1}^{j^{m+1}T}X_{\tau+1}^{j^{m+1}}$, $s_{\tau+1}^{m+1} = X_{\tau+1}^{j^{m+1}T}U_{\tau+1}^{m+1}$,
   $\widehat{\beta}^{m+1} = (P_\tau^{m+1} + P_{\tau+1}^{m+1})^{-1}(s_\tau^{m+1} + s_{\tau+1}^{m+1})$.
   $\widehat{f}_{m+1} = X_{\tau+1}\widehat{\beta}^{m+1}$, $\tilde{F}_{m+1} = \widehat{F}_m + \widehat{f}_{m+1}$, $\widehat{F}_{m+1} = sign(\tilde{F}_{m+1})\min(1, |\tilde{F}_{m+1}|)$.
4) **Iteration** If $m + 1 < M$ increase m by 1 and goto step3. If $m + 1 = M$ goto step2.

The online L_2 with temporal models is an approximation to the batch version of L_2Boost of Section II, due to the incremental computation of the residuals of the step 3. This approximation allows us to update the linear models without storing any of the data points. Furthermore, we intentionally use the same set of features learned offline in order to be able to perform this update online and during run time.

B. Unsupervised adaptation

We aim to update the temporal models of the L_2boost in order to tune the classifiers to work correctly in the new environments it's deployed. Thus, we aim to adapt every new data point acquired into the model using the online framework just described, in difference to previous approaches that discard [7], or select points [2], [1] at every adaptation step. We address the problem in two separate steps: (i) Offline-supervised learning, that outputs a classifier *prototype* consisting of one set of features, Θ_j, and one set of temporal models, Θ_β^τ, and (ii) ongoing unsupervised update of the temporal models Θ_β^τ.

Then, the adaptation step updates the parameters by taking into account the new data points, $X_{i,\tau+1}$, $i = 1, \ldots, n$, generating the updated temporal models $\Theta_\beta^{\tau+1}$, from the following unsupervised cost function:

$$\hat{\beta}_{\tau+1} = \arg\min_\beta \sum_{i=1}^{n} \gamma_i(\hat{y}_{i,\tau+1} - F(X_{i,\tau+1}; \beta))^2. \quad (4)$$

Where $\hat{y}_{i,\tau+1} = \operatorname{sgn}(F_M(X_{i,\tau+1}; \Theta_j, \Theta_\beta^\tau))$ is the pseudolabel for the new unlabeled point, $X_{i,\tau+1}$, and $\gamma_i = |F_M(X_{i,\tau+1}; \Theta_j, \Theta_\beta^\tau)|$ the probability of that point belonging to that class (i.e. margin-based confidence). Note that the class probability works as a weighting factor in the optimization and that both parameters are computed using the previous classifier, $\{\Theta_j, \Theta_\beta^\tau\}$.

To use matrix notation, consider Γ as a diagonal matrix with entries $\Gamma_{ii} = \sqrt{\gamma_i}$ and $U_{\tau+1}$ as a vector that contains the objective values for all the new points (note that this vector represents the pseudolabel in the first round and the residuals in the following ones). Eq. (4) solves, at each round, a weighted least squares problem as follows:

$$\begin{aligned}\beta_{\tau+1}^m &= \arg\min_\beta (U_{\tau+1} - X_{\tau+1}^{j^m}\beta)^T\Gamma^2(U_{\tau+1} - X_{\tau+1}^{j^m}\beta) \quad (5)\\ &= ((\Gamma X_{\tau+1}^{j^m})^T(\Gamma X_{\tau+1}^{j^m}))^{-1}(\Gamma X_{\tau+1}^{j^m})^T\Gamma U_{\tau+1}\end{aligned}$$

In order to use the online algorithm described in the previous section with unlabeled data, $(X_{\tau+1})$, three changes must be made: i) substitute $X^{j^m}_{\tau+1}$ for $\Gamma X^{j^m}_{\tau+1}$, ii) substitute $U^m_{\tau+1}$ for $\Gamma U^m_{\tau+1}$ and iii) use the pseudolabel vector $\hat{Y}_{\tau+1}$ instead of the nonexistent label Y.

C. Multi-class update

On binary problems, the weights γ_i compute the probability of classifying one class vs. other class (Eq. (2)). In order to extend the regularization weights Γ to multi-class problems we use the one vs. all approach, which solves C binary problems to discriminate between C classes where $Y \in \{1, \ldots, C\}$. The multi-class version of L_2 starts by computing $\widehat{F}^{(c)}_M$ on the basis of the binary response variables

$$Y^{(c)}_i = \begin{cases} 1 & \text{if } Y_i = c \\ -1 & \text{if } Y_i \neq c \end{cases} i = 1, \ldots, n \tag{6}$$

and then builds the classifier as $\widehat{C}^m(x) = \arg\max_{c\in\{1,\ldots,C\}} \widehat{F}^{(c)}_M(x)$.

After the arrival of new data $X_{\tau+1}$, we need to adapt the C binary classifiers in order to update the multi-class model.In addition, we must compute weights that consider the classification probability of a multi-class problem. In difference to the binary case that uses the two available classes, in multi-class problems we have c weights, one from each binary problem. For each point the multi-class weight is computed considering two cases:

$$\gamma^{(c)}_i = \begin{cases} |\widehat{F}^{(\widehat{C}^m(x))}(x) - \widehat{F}^{(\widehat{C}^m_1(x))}(x)|/2 & \text{if } c = \widehat{C}^m \\ |\widehat{F}^{(\widehat{C}^m(x))}(x) - \widehat{F}^{(c)}(x)|/2 & \text{if } c \neq \widehat{C}^m \end{cases} \tag{7}$$

$$\widehat{C}^m_1(x) = \arg \max_{c\in\{1,\ldots,C\}\wedge c\neq\widehat{C}^m(x)} \widehat{F}^{(c)}(x),$$

where $\widehat{C}^m_1$ is the second most probable class label of point x. In practice, only if one have a probability of more than 50% one can have some confidence that the new point is correctly classified.

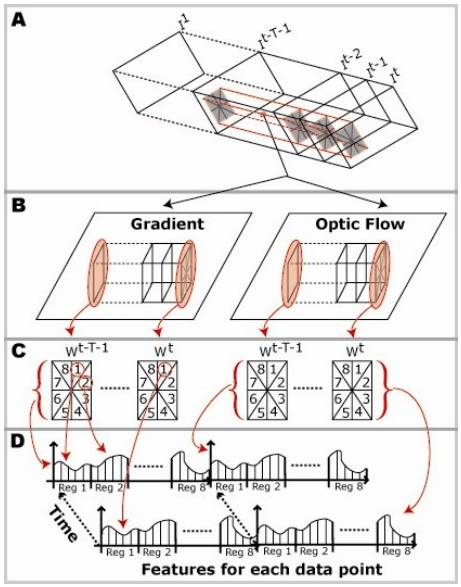
IV. Online adaptive human actions classifier

In order to assess the adaptation of the L_2boost we recorded sequences in two different scenarios with slightly different conditions and the same activities recorded in the Weizmann data set.

The sequences of the first scenario just add a non-uniform background to the activities, while the sequences of the second scenario are recorded in a very textured background with several shadows and several amounts of motion blur. The objective of the first scenario is to evaluate the performance of the adaptation quantitatively, while the objective of the second scenario is to check the limits of the adaptation procedure. Example images of the three scenarios are presented in Figure 2(a).The actions used are: {1 - bending down, 2 - jumping jack, 3 - jumping, 4 - jumping in place, 5 - running, 6 - galloping sideways, 7 - walking, 8 - waving one hand, 9 - waving both hands}. [1]

[1]The sequences used, the ground truth and the code used can be found on the web: ual2boost.wikispaces.com

(a)



(b)

Figure 2. (2(a)) Example image for the three setups (top row) used in the experiments: Weizmann (top left), IST1 (top middle) and IST2 (top right). Rows 2,3 and 4 show zoomed examples of the action run for all the three scenarios with an interval of four frames (2(b)) Feature computation: A) example of a volume of video used to compute the features for the person detected in image I^t, B) the two types of raw features used, gradient and flow vectors, computed inside the volume correspondent to the person detected, C) polar sampling used to divide each window into subregions and D) weighted histograms computed for each region, producing a 2D matrix coding the evolution of each bin over a set of T frames.

A. Feature computation

The state-of-the-art action recognition approaches use a combination of appearance and motion-based features in order to extract the activities' patterns from videos [11]. We follow this approach, using the image gradient and optical flow (dense) as the raw features to extract the action patterns. Figure 2(b)-A and 2(b)-B show an example of the video volume for feature computation. Note that the bounding box maintain the same location over the frames selected, so we do not apply any person tracking or segmentation algorithm before the feature computation.Given a gradient

Offline (Weizmann)				**UAL$_2$Boost**	Semi-boost
Feats	Classf	Test		**Test**	Test
		Weiz.	IST1	**IST1**	IST1
PCA[11]	SVM [11]	99,6 (99,6)	75,4 (77,5)	-	55,3 (56,7)
Ours	L2boost	94,8 (97)	73,5 (73,7)	**89,7 (90,5)**	84,1 (85,9)

Table I
RESULTS OF IST1 SCENARIO: TRACE OF THE CONFUSION MATRIX (TRUE RECOGNITION RATE).

image or optic flow image, the weighed histogram divides the image in subregions (according to a sampling strategy, e.g. cartesian, polar) and computes the histogram of the gradient (or flow) orientation weighted by its magnitude. Figure 2(b)-C shows the polar sampling strategy, which shown better discrimination capabilities. The histogram features are parametrized by the number of subregions nR and the number of bins nB of the histogram. We denote the gradient histogram as $g^t \in \mathbb{R}^{nB \cdot nR}$ and the flow histograms as $o^t \in \mathbb{R}^{nB \cdot nR}$, computed at frame t. For all the experiments the number of bins of the polar sampling are $nR = 16$, $nB = 16$, obtained from an initial parameter selection stage.

We compute the gradient and flow weighted histograms in consecutive frames in order to extract a "volume" of histograms that encode the temporal evolution of the features, as shown in Figure 2(b)-D. The gradient and optic flow histogram volume of size T corresponds to the action model, $\mathcal{H} = \{g^t, o^t\}$, $t = 1, \ldots, T$. The size of the temporal model is $T = 10$, obtained from an initial parameter selection stage. At frame t, the appearance and motion feature vector for each person detected is $h^t = [g^t o^t]^T \in \mathbb{R}^{2 \cdot nB \cdot nR}$. We define the component h_j^t in the previous T frames as the feature vector of the L$_2$boost. Thus, each feature vector is $X_i^j = [h_j^t h_j^{t-1} \cdots h_j^{t-T-1} 1] \in \mathbb{R}^{T+1}$, and the L$_2$Boost selects the component $j = \{1, \ldots, 2 \cdot nB \cdot nR\}$ and the linear parameters β that minimizes the cost function in Eq. 1.

B. Experiments

We compare of our approach for feature computation and unsupervised adaptation with: (i) One of the recent approaches for the supervised learning of human activities in the Weizmann data set and (ii) the semi-boost learning algorithm, with parameters: rounds $T = 30$, $\sigma = 97$th percentile, selecting the top 10% samples [1]. The performance of the multi-class learning on the testing set is evaluated with two criteria: (i) The mean value of the trace of the confusion matrix and (ii) the recognition rate over all the samples. The adaptation procedure is performed sequentially for each new data sample.

1) Adaptable scenario - IST1: The IST1 scenario contains the same actions of the Weizmann data set, performed by nine subjects. The differences of IST1 with respect to the Weizmann setup are: (i) a non-uniform background, (ii) the subject's size is smaller with respect to the image size and (iii) the presence of motion blur in some actions. These slight changes will allow the unsupervised adaptation to update the temporal models in order to improve the performance on the IST1 scenario after the adaptation. The table I shows the results of the test performed on the IST1 scenario.

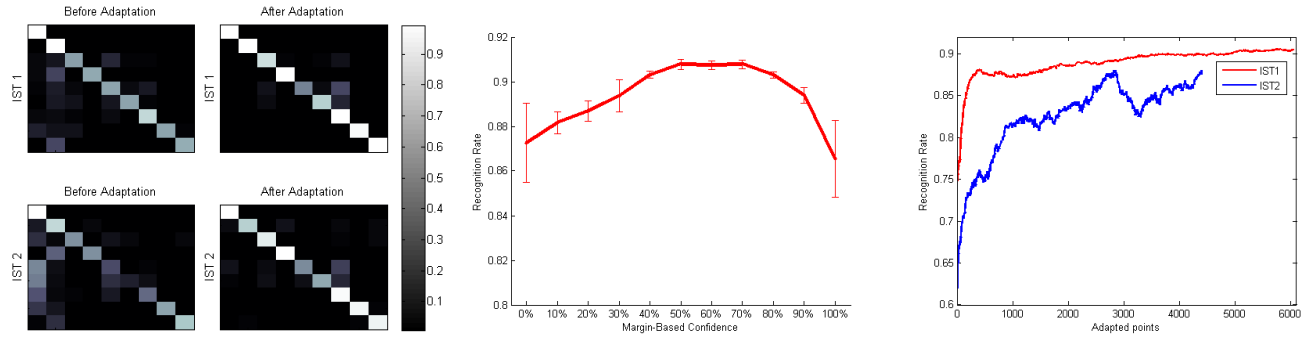
We observe in Table I that the *UAL$_2$Boost* improves the recognition rate when compared to the L$_2$Boost offline classifier. In addition, our approach attain better classification results than the semi-boost algorithm. It is important to remark that our approach works on-line, while semi-boost works offline with all the data (Weizmann + IST1). Thus, our offline approach has selected the appropriate features Θ_j that allow an online model adaptation.

We show the detailed results of the confusion matrix computed before and after the adaptation in Figure 3(a), remarking that in almost all actions the algorithm was able to improve the performance. Figure 3(b) shows the recognition rate after adaptation for several confidence thresholds (i.e., if $\gamma_i^{(c)} \geq t$ then perform adaptation of (7)), considering 50 different shuffles of the IST1 data samples. The curve shows that a confidence threshold between 50% and 70% provides good adaptation results. Figure 3(c) shows the evolution of the recognition rate over the selected data points, with $\gamma_i^{(c)} \geq 0.5$, which is the minimum probability we should attain to consider the new point correctly classified.

2) Non-adaptable scenario - IST2: The IST2 scenario contains the same actions of the Weizmann data set, performed by three subjects. The design of the new scenarios (IST1 and IST2) was based on the incremental complexity levels, so the IST1 is more complex than Weizmann and IST2 is more complex than IST1. This means that we should be able to obtain better classification results on IST2 by performing: (i) L$_2$Boost offline on the Weizmann data set, (ii) UAL$_2$Boost on IST1. Table II shows that the incremental adaptation performs much better than the semi-boost applied on the union of all data sets.

These results clearly suggest that it is possible to create self tunning classifiers that adapt in an incremental way to more complex situations. This kind of learning paradigm can be viewed as starting from simple cases and gradually moving towards more complex ones. Note also the performance gap between the semi-boost with L$_2$Boost and UAL$_2$Boost.

Figure 3(c) shows the evolution of the recognition rate over the IST2 data set along the adaptation of the selected point from IST1, showing an improvement form 62% to 88%. It is important to remark that our method is not able to adapt the classifier learnt offline on the Weizmann scenario directly to the IST2 scenario, due to the large differences on the sequences. Nevertheless, the approach of *from simple to complex* is able to improve the performance of the classifier on the IST2 scenario in an unsupervised fashion.



(a) (b) (c)

Figure 3. (3(a)) On the top row, the confusion matrices for the IST1 setup before and after adaptation. On the bottom row, the confusion matrices for the IST2, before and after adaptation. (3(b)) shows the recognition rate over all data for different values of the confidence threshold for the IST1 scenario. (3(c)) shows the evolution of the recognition rate over the selected data points

Our features + L2boost	IST2
Offline (Weizmann) + UAL$_2$Boost (IST1)	**86,2(88,0)**
Semi-boost (Weizmann + IST1 + IST2)	55,6 (54,6)

Table II
RESULTS OF WEIZMANN + IST1 SCENARIO.

V. CONCLUSIONS

We introduce a learning algorithm that is able to perform automatic update of the models of the L2boost procedure in multi-class problems. Our algorithm performs an unsupervised and on-line adaptation of the new data samples in two stages: (i) the offline initialization of the linear temporal models of L2boost and (ii) the online unsupervised adaptation of the models. The adaptation step relies on: (a) the pseudo-labels and (b) the margin-based confidence weights. These elements allow to: (i) improve the classification performance of unlabeled data and (ii) include all the new samples during the adaptation.

We apply the algorithm on the unsupervised adaptation of models for human action recognition and evaluate the algorithm in two scenarios: (i) an adaptable scenario, which introduces minor changes to the patterns trained offline, and (ii) a non-adaptable scenario, which introduces large changes that generally need the retraining of the classifier. The experimental results demonstrate that our algorithm is able to: (i) attain very good classification rates in the adaptable scenario and (ii) improve the classification rates in the non-adaptable scenario following a *from simple to complex* approach. In addition, our algorithm obtains better results than a batch state-of-the-art semi-supervised method, the semi-boost algorithm [1], and clearly outperforms the current state-of-the-art supervised method [11], in this database, with respect to its applicability to other similar scenarios.

Further analysis of our approach should explore the conditions that guarantee convergence to a better classification result and test the system during very long periods of time.

ACKNOWLEDGMENT

This work was supported by FCT (ISR/IST plurianual funding through the PIDDAC Program) and partially funded by EU Project First-MM (FP7-ICT-248258) and EU Project HANDLE (FP7-ICT-231640). Pedro Ribeiro also acknowledges the FCT PhD grant SFRH/BD/18936/2004.

REFERENCES

[1] P. K. Mallapragada, R. Jin, A. K. Jain, and Y. Liu, "Semi-boost: Boosting for semi-supervised learning," *IEEE Transactions on PAMI*, vol. 31, no. 11, pp. 2000–2014, 2009.

[2] K. P. Bennett, A. Demiriz, and R. Maclin, "Exploiting unlabeled data in ensemble methods," in *ACM SIGKDD02, International conference on*.

[3] T. Parag, F. Porikli, and A. Elgammal, "Boosting adaptive linear weak classifiers for online learning and tracking," in *CVPR08*.

[4] C. Leistner, H. Grabner, and H. Bischof, "Semi-supervised boosting using visual similarity learning," in *CVPR08*.

[5] N. C. Oza and S. Russell, "Online bagging and boosting," in *Eighth International Workshop on Artificial Intelligence and Statistics*, January.

[6] H. Grabner and H. Bischof, "On-line boosting and vision," in *CVPR06*.

[7] O. Javed, S. Ali, and M. Shah, "Online detection and classification of moving objects using progressively improving detectors," in *CVPR05*.

[8] C. Huang, H. Ai, T. Yamashita, S. Lao, and M. Kawade, "Incremental learning of boosted face detector."

[9] B. Wu and R. Nevatia, "Improving part based object detection by unsupervised, online boosting."

[10] P. Buhlmann and B. Yu, "Boosting with the l2 loss: Regression and classification," *Journal of the American Statistical Association*, vol. 98, pp. 324–339, January 2003.

[11] K. Schindler and L. van Gool, "Action snippets: How many frames does human action recognition require?" in *CVPR08*.