Applied Soft Computing 8 (2008) 1314–1325

Contents lists available at ScienceDirect

Applied Soft Computing

journal homepage: www.elsevier.com/locate/asoc

Dynamic classification for video stream using support vector machine

Mariette Awad [a,b,*], Yuichi Motai [a]

[a] Department of Electrical and Computer Engineering, University of Vermont, Burlington, VT, USA
[b] IBM Systems and Technology Group, Essex Junction, VT, USA

ARTICLE INFO

Article history:
Received 8 October 2007
Accepted 8 November 2007
Available online 24 April 2008

Keywords:
Dynamic soft computing
Multiple classification
Incremental support vector machine
Behavior learning

ABSTRACT

A dynamic classification using the support vector machine (SVM) technique is presented in this paper as a new 'incremental' framework for multiple-classifying video stream data. The contribution of this study is the derivation of a unique, fast and simple to implement technique that allows multi-classification of behavioral motions based on an adaptation of the least-square SVM (LS-SVM) formulation. This dynamic approach leads to an extension of SVM beyond its current static image-based learning capabilities. The proposed incremental multi-classification method is applied to video stream data, which consists of an articulated humanoid model monitored by a surveillance camera. The initial supervised off-line learning phase is followed by a visual behavior data acquisition and then an incremental learning phase. The resulting error rate and the confidence level for the proposed technique demonstrate its validity and merits in articulated motion learning. Furthermore, the enabled online learning allows an adaptive domain knowledge insertion and provides the advantage of reducing both the model training time and the information storage requirements of the overall system which are both essential for dynamic soft computing applications.

1. Introduction

Dynamic soft computing represents a significant paradigm shift in the strict definition of computing. Unlike conventional (hard) computing, it demonstrates the feasibility of storing and processing information in a manner tolerant of imprecision, uncertainty and approximation. The basic ideas underlying the standard soft computing incarnation were influenced by Zadeh's work on fuzzy sets, analysis of complex systems and decision processes, as well as the possibility theory and soft data analysis. Soft Computing (SC) has Fuzzy Logic (FL), Neural Computing (NC), Evolutionary Computation (EC), Machine Learning (ML) and Probabilistic Reasoning (PR) as principal constituents [14,2].

Dynamic classification is a desirable method in ML applications and it serves as a basis for future decision-making [14]. Researchers in classification had proposed many different algorithms such as k-nearest-neighbor (KNN), decision tree induction, error propagation, reinforcement learning, lazy learning, rule based and statistical learning algorithms [33]. Dynamic refers, in this context, to the situation where the training dataset is not fully available at the beginning of the learning process. Hereafter, we will interchangeably use dynamic, incremental or online learning to refer to the patterns that become available and need to be incorporated into the already computed ML model to preserve the class concept.

Our study focuses on SVM as a prime classifier for an incremental multi-classification mechanism applied to sequential stream video which is an active area for further research [5]. SVM is a new learning by example paradigm spanning a wide spectrum of tasks ranging from classification, regression to density estimation problems using supervised, semi-supervised and unsupervised type of learning [14]. Our selection for SVM among the numerous learning algorithms available in the literature is justified by several of its main advantages: SVM is computationally efficient, highly resistant to noisy data and offers powerful generalization capabilities on yet-to-be-seen data [11]. These characteristics make SVM an attractive target for soft computing since the guiding principle of soft computing is to exploit the tolerance for imprecision and partial knowledge to develop robust and efficient solutions [2]. Because SVM formulation embodies the structural risk minimization (SRM) which is provided as the probability of misclassifying unseen data using maximum margin between the separating hyper planes and not the traditional empirical risk minimization (ERM) for misclassifying the training data, it distinguishes itself from neural networks, linear discriminant analysis (LDA), KNN and other non-parametric techniques by the fact that it does not exhibit the same classical problems of multi-

* Corresponding author at: Department of Electrical and Computer Engineering, University of Vermont, Burlington, VT, USA. Tel.: +1 802 769 2923.
E-mail address: mawad@us.ibm.com (M. Awad).

1568-4946/$ – see front matter © 2008 Elsevier B.V. All rights reserved.
doi:10.1016/j.asoc.2007.11.008

local minima, curse of dimensionality and over-fitting [35]. This makes SVM suitable for the specific case of video stream data which can be noisy and exhibit a large degree of variability in the object class and the environment – such as occlusions, orientation and illumination of the object – between the training and testing phases of developing the classifier model [9].

Constructing a ML SVM classifier capable of incremental classification as opposed to batch-mode learning is very attractive and will become a strategic necessity for video stream data for many reasons. First there is a significant demand increase in video stream applications such as smart surveillance [6], building a profile of people manners [17,29], remote monitoring of elderly patients in healthcare centers, elucidating rodent behavior under different conditions [15] as well as articulated motion analysis, visual-based human and computer interface. Second real time classification of video stream imposes extensive computational requirements on the system during training and even at run time [9]. Third video stream data have time dependent characteristics or are obtained at intervals because of their nature. This usually requires an extensive training phase for the initial classifier based on a large off-line collection of datasets that represent the learning-from-examples paradigm. If the incorporation of incremental modular algorithms for video stream classification could be designed to meet low storage and memory requirements, online learning will reduce the extensive time consuming and resource extensive machine learning training phase without affecting the classifier ability to continuously learn. This would enable dynamic soft computing and maintain good fit for the initially sparse training data as well as adequate prediction capabilities on the incremental data that were not included in the training samples.

Motivated by the main goals of ML to construct a machine that learns facts through exploration, improves cognitive skills through practice and organizes new knowledge into effective representation, we present a novel technique that extends traditional SVM beyond its existing static image-based learning capabilities to handle multiple behavioral classifications. We opted to investigate dynamic behavior learning because of the numerous current and potential applications mentioned earlier. For illustration purposes, we have applied our technique to learn the behavior of an articulated humanoid through video footage captured by a monitoring camera sensor. We have then tested the model's accuracy of incrementally classifying articulated motions. The initial supervised off-line learning phase was followed by a visual behavior data acquisition and then an online learning phase.

To the best of our knowledge, no prior work has used an adaptation of LS-SVM with a multi-classification objective for dynamic behavior learning. The contribution of this study is the derivation of this unique, dynamic, fast and simple to implement multi-classification technique which satisfies the principles of soft computing.

This paper is organized as follows. Section 2 presents an overview of SVM principles and related techniques. Section 3 covers our unique multi-classification procedure and Section 4 introduces our proposed incremental SVM. Section 5 describes the experimental set-up and Section 6 summarizes the experimental analysis and results obtained. Finally, Section 7 contains concluding remarks and outlines our plans for follow-on work.

2. SVM principles and related studies

Originally designed for binary classification, the SVM techniques were invented by Boser, Guyon and Vapnik and were introduced during the Computational Learning Theory (COLT) conference of 1992 [11]. SVM offers a principled approach to ML problems because of its mathematical foundations in statistical learning theory [35]. The aim of SVM classification is to find a computationally efficient way of learning a hyper plane that correctly classifies the high dimensional feature space. SVM tries to minimize the confidence interval and keep the training error fixed while maximizing the distance between the calculated hyper plane and the nearest data points known as support vectors (SVs). SV define the margins for the hyper planes and summarize the remaining data, which can then be ignored. The complexity of the classification task will thus depend on the number of support vectors rather than on the dimensionality of the input space and this helps prevent over-fitting. Calculating SV and specifically controlling their selection is one of the major limitations of SVM techniques [5].

Traditionally, SVM was considered for unsupervised off-line batch computation, binary classifications, regressions and structural risk minimization (SRM) [14]. Adaptations of SVM were applied to density estimation [28], Bayes point estimation [8] and transduction [14] problems. Researchers also extended the SVM concepts to address error margin, efficiency, multi-classification and incremental learning. Platt [30] introduced iterative chunking of the input space by heuristically selecting an 'active or working' set to train SVM using generic optimization techniques. In his approach, the support vectors from the chunk are retained and the remaining part of the data is tested against the hypothesis found. All points that drastically violate the Karush–Kuhn–Tucker (KKT) conditions are added to the support vectors of the previous problem forming a new chunk to process. For non-sparse problems or for a large set of support vectors, this technique is not very attractive because its heuristic nature requires the set of support vectors to be stored in system memory before being processed by the optimization algorithm. To improve efficiency and to end the need for iterative chunking in large data sets, Suykens and Vandewalle [34] developed a least-squares SVM (LS-SVM) technique by removing the inequality constraints from the traditional Vapnik SVM quadratic formulation. The optimization problem is thus reduced to a set of linear equations with KKT conditions to be satisfied. With LS-SVM, the sparseness of the solution is basically lost and the best support vectors estimation occurs when error variables fit a Gaussian distribution curve. d'Alche'-Buc and Ralaivola [12] introduced an incremental learning algorithm that exploits the locality of Radial Basis Function (RBF) by re-learning only the weights of the training data points lying in the vicinity of the new incremental data. Their study however did not address any drifting concepts. Cauwenberghs and Poggio [10] presented an online recursive training algorithm that allowed incrementing and decrementing to enable an evaluation for the 'leave one out' generalization performance. Despite the fact that the 'leave one out' error is almost unbiased and can be proved by the Luntz–Brailovsky theorem, computing it is expensive in general because training on L sets of size $(L-1)$ is required.

From a mathematical perspective, SVM hyper planes for binary classification are given by: $wx_i + b = 0$. Suppose $T = \{(x_1,y_1), \ldots, (x_N,y_N)\}$, where $x_i \in R^n$ is a training set with *attributes or features* $\langle f_1, f_2, \ldots, f_n\rangle$ and $T^+ = \{x_i|(x_i,y_i) \in T \text{ and } y_i = 1\}$ and $T^- = \{x_i|(x_i,y_i) \in T \text{ and } y_i = -1\}$ be the set of positive and negative training examples, respectively. For a correct classification, all x_i's must satisfy: $y_i(wx_i + b) \geq 0$. Among all such planes satisfying this condition, SVM finds the optimal hyper plane P_0 where the margin distance between the decision plane and the closest sample points is maximal. P_0 is defined by its slope w and should be situated equidistant from the closest point on either side. Let P_+ and P_- be two additional planes that are parallel to P_0 and include the support vectors. P_+ and P_- are defined respectively by: $wx_i + b = 1$, $wx_i + b = -1$. All points x_i should satisfy $wx_i + b \geq 1$ for $y_i = 1$, or $wx_i + b \leq 1$ for $y_i = -1$. Thus combining the

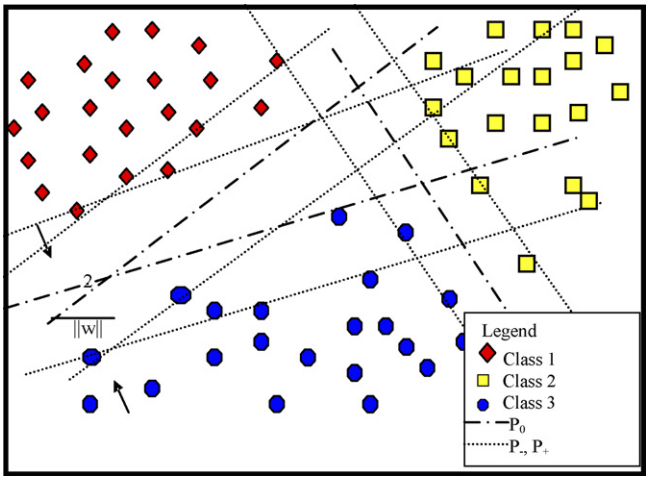


Fig. 1. Standard multi-class SVM.

conditions for all points x_i we have: $y_i(wx_i + b) \geq 1$. The distances from the origin to the three planes P_0, P_+ and P_- are respectively, $|b-1|/||w||, |b|/||w||$ and $|b+1|/||w||$ [10]. Fig. 1 represents the positions of the three planes for a multi-classification case.

Eq. (1) through Eq. (6) presented below are based on Forsyth and Ponce [18] and are applicable for a binary and linearly separable classification. The optimal plane needs to minimize the quadratic objective function of Eq. (1) subject to the constraint in Eq. (2). As mentioned earlier, this optimal plane is the same as the hyper plane for which the separation margin between the two classes (measured along a perpendicular to the hyper plane) is also maximized:

$$\text{Objective function}: \frac{1}{2}w^{\mathrm{T}}w \tag{1}$$

$$\text{Constraint linearly separable case}: y_i(w^{\mathrm{T}}x_i + b) \geq 1 \tag{2}$$

Since the objective function is quadratic, this constrained optimization is a quadratic programming (QP) task and is solved by Lagrange multipliers method. The goal is to minimize the Lagrange expression L_{p} with respect to w, b and the Lagrange coefficients $\alpha_i \geq 0$ where:

$$L_P(w,b,\alpha) = \frac{1}{2}w^{\mathrm{T}}w - \sum_{i=1}^{N}\alpha_i(y_i(w^T x_i + b) - 1) \tag{3}$$

By setting the partial derivatives of Eq. (3) with respect to w and b equal to 0:

$$\frac{\partial}{\partial w}L_{\mathrm{p}}(w,b) = 0, \quad \frac{\partial}{\partial b}L_{\mathrm{p}}(w,b) = 0$$

We get:

$$w = \sum_{j=1}^{N}\alpha_j y_j x_j \quad \text{and} \quad \sum_{j=1}^{N}\alpha_j y_j = 0 \tag{4}$$

Substituting Eq. (4) back into the Lagrange expression, we now deal with the dual Lagrange problem of the above primal. Both problems arise from the same objective function but they have different constraints. In the dual Lagrange, the solution is found by maximize L_{p} whereas in the primal Lagrange, the solution is found by minimizing L_{d}:

$$L_{\mathrm{d}}(\alpha) = \sum_{i=1}^{N}\alpha_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j y_i y_j (x_i^{\mathrm{T}}x_j)$$

Any new data point is then classified by the decision function in Eq. (5):

$$\text{Decision function}: f(x) = \text{sign}(w^{\mathrm{T}}x + b) \tag{5}$$

Substituting Eq. (4) into Eq. (5) allows us to rewrite the decision function as

$$f(x) = \text{sign}(w^{\mathrm{T}}x + b) = \text{sign}\left(\sum_{i=1}^{N}\alpha_i y_i x^{\mathrm{T}}x_i + b\right) \tag{6}$$

For most practical applications where the class data are not completely separable, a slack variable e is added to Eq. (2) so that the hyper plane maximizes the margin between the two classes while minimizing e which is proportional to the misclassification error [16]:

$$\text{Constraint for non-separable case}: y_i(w^{\mathrm{T}}x_i + b) \geq 1 - e_i$$

SVM solutions are obtained from solving quadratic programming (QP) problems. LS-SVM instead uses linear system instead of QP and as introduced in [34] are obtained by optimizing the Lagrangian as defined in Eq. (7):

$$L_{\mathrm{p}}(w,b,e,\alpha) = \frac{1}{2}w^{\mathrm{T}}w + \frac{1}{2}\lambda\sum_{i=1}^{N}(e_i^2) - \sum_{i=1}^{N}\alpha_i(y_i(w^{\mathrm{T}}x_i + b) - 1 + e_i) \tag{7}$$

with α_i being the Lagrange multipliers which can be either positive or negative. These parameters are derived from Karuch–Kuhn–Tucker (KKT) conditions which are valid as long as the objective function and conditions are convex [34]. The gradient of the inequality restrain solutions to the interior of the acceptable region and the optimal solution is the saddle point that satisfies Eq. (4). λ is the regulating parameter for the error term which greatly impacts the classifier performance during the training phase. Hsu and Lin [22] showed that SVM accuracy rates were influenced by the selection of λ, which varies in ranges depending on the problem under investigation. KKT conditions:

$$\begin{cases} \dfrac{\delta L}{\delta w} = 0 \rightarrow w = \displaystyle\sum_{i=1}^{N}\alpha_i y_i x_i \\ \dfrac{\delta L}{\delta b} = 0 \rightarrow \displaystyle\sum_{i=1}^{N}\alpha_i y_i = 0 \\ \dfrac{\delta L}{\delta e_i} = 0 \rightarrow \alpha_i = \lambda e_i \\ \dfrac{\delta L}{\delta \alpha_i} = 0 \rightarrow y_i(w^{\mathrm{T}}x_i + b) - 1 + e_i = 0 \end{cases} \tag{8}$$

Eliminating w and e, the system of linear equations in Eq. (8) can be written more concisely in a matrix form as

$$\begin{bmatrix} 0 & Y^T \\ Y & ZZ^{\mathrm{T}} + \lambda^{-1}I \end{bmatrix}\begin{bmatrix} b \\ \alpha \end{bmatrix} = \begin{bmatrix} 0 \\ I \end{bmatrix}$$

where

$$\begin{cases} Z = [x_1^{\mathrm{T}}y_1, \ldots, x_N^{\mathrm{T}}y_N] \\ Y = [y_1, \ldots, y_N] \\ I = [1, \ldots, 1] \\ e = [e_1, \ldots, e_N] \\ \alpha = [\alpha_1, \ldots, \alpha_N] \end{cases}$$

with Mercer conditions applied to ZZ^{T} [34].

When data cannot be classified by linear decision surfaces, Cover's theorem states that the input space may be transformed into a new feature space that is linear separable [7]. Non-linear transformation called inner-product kernels help in mapping the

input space to a high dimensional feature space. Most common kernels used are [18]:

$$\text{Linear} : k(x, x_i) = x^{\mathrm{T}} x_i$$

$$\text{Polynomial} : k(x, x_i) = (x^{\mathrm{T}} x_i + 1)^d$$

$$\text{Gaussian(Radial-basis)} : k(x, x_i) = \exp\left(\frac{-||x - x_i||^2}{2\sigma^2}\right)$$

$$\text{Sigmoid} : k(x, x_i) = \tanh(\beta_0 x^{\mathrm{T}} x_i + \beta_1)$$

SVM performance largely depends on the choice of kernels. However, there are no well-established theories to select kernel functions in a data-dependant way. Amari and Wu [1] proposed a modified kernel which takes into consideration geometric information of the Riemannian structure and enlarges the spatial resolution to increase class separability.

There are different strategies that researchers used to decompose the multi-classification problem into a set of binary ones. Schölkopf and Smola [32] described the widely used multi-classification SVM techniques: one-versus-rest, pair-wise classification, error-correcting output codes and the multi-classification objective functions. The one-versus-the-rest also referred to as one-against-all (OAA) is probably the earliest SVM multi-class implementation [4]. It constructs c binary SVM classifiers where c is the number of classes. Each one distinguishes one class from all the other classes which reduces the case to a two-class problem. There are c decision functions: $w_1^{\mathrm{T}} x_i + b_1, \cdots, w_c^{\mathrm{T}} x_i + b_c$. The final label output is given to the class that showed the highest output value [22].

$$\text{Class of } x \equiv \text{argmax}_{i=1,\cdots,c}(w_i^{\mathrm{T}} x + b_i) \tag{9}$$

The pair-wise classification also referred to as one-against-one (OAO) builds $c(c-1)/2$ binary SVM, each of which is used to discriminate two of the c classes only. For the kth and jth class training data, the constraints are [18]:

$$w_{kj}^{\mathrm{T}} x_i + b_{kj} \geq 1 - e_{kj}^i \quad \text{for } y_i = k$$

$$w_{kj}^{\mathrm{T}} x_i + b_{kj} \geq -1 - e_{kj}^i \quad \text{for } y_i = j$$

The one-to-others approach trains c SVM versus c^2 however at runtime both techniques require the evaluation of $(c-1)$ SVM classifiers [32]. The multi-classification objective function has probably the most compact form as it optimize the problem in one single step. It constructs c two class rules, where each classifier separates training vectors of a class from all the others using the constraint:

$$w_{y_i}^{\mathrm{T}} x_i + b_{y_i} \geq w_m^{\mathrm{T}} x_i + b_m - 2 - e_i^m$$

The decision function is the same as Eq. (9) for the one-against all technique [22].

For reasonable data set sizes, the accuracy of the different multi-classification techniques is comparable. For any particular problem, the selection of the optimal multi-classification approach partly depends on the required accuracy, development and training time.

3. Proposed multi-classification SVM

Using a similar approach to Hsu and Lin [22], we add the plane intercept term b to the objective function:

$$\frac{1}{2}\sum_{m=1}^{c}(w_m^{\mathrm{T}} w_m + b_m b_m) + \lambda \sum_{i=1}^{N}\sum_{m \neq y_i}^{c}(e_i^m)^2 \tag{10}$$

Adding b into the objective function that already accounts for w enables us to uniquely define the optimal plane. The selection of λ can be found heuristically or by a grid search. Large λ values favor less smooth solutions that drive large w values.

Selecting the multi-classification objective function among the different schemes of multi-classification, we use the constraint as shown in Eq. (11) with a slack variable to account for the non-separable data:

$$(w_{y_i}^{\mathrm{T}} x_i) + b_{y_i} = (w_m^{\mathrm{T}} x_i) + b_m + 2 - e_i^m \tag{11}$$

However, unlike traditional multi-class SVM and LS-SVM, we consider the constraint relationship to be equality instead of inequality between the different classes. Substituting Eq. (11) into Eq. (10), we drop the Lagrange multipliers and get the following objective function to optimize:

$$L(w, b) = \frac{1}{2}\sum_{m=1}^{c}(w_m w_m + b_m b_m) + \frac{\lambda}{2}\sum_{i=1}^{N}\sum_{m \neq y_i}^{c}((w_{y_i} - w_m)x_i + (b_{y_i} - b_m) - 2)^2$$

The objective function remains convex which insures that the KKT conditions are still valid, but the model is now reduced to a single system of linear equations. The uniqueness of the global solution in our proposal is still valid because it is a property of the Hessian being positive definite or semi-definite [16]. Fundamentally the problem is now different than Eq. (7). It is reduced to an unconstrained optimization and the solution now becomes equal to the rate of change in the value of the objective function. We do not numerically solve for SV that correspond to the nonzero Lagrange multipliers in traditional SVM. Instead our solution now classifies points by assigning them to the closest parallel planes without explicitly calculating SV. The hyper planes are pushed apart by a maximum margin and points of each class are now clustered around them with no input data lying on planes. Fig. 2 represents a geometrical illustration for the proposed approach. The elimination of SV calculation and the Lagrange multipliers makes our technique faster than standard SVM which is known to converge slower than neural networks for a given generalization performance [19,20]. Our proposed technique is also easy to implement and as in soft computing, by not exactly solving for SV, it exploits the tolerance for imprecision and uncertainty to achieve acceptable misclassification error for the dynamic multi-classification SVM as detailed in Section 4.

In traditional SVM, the order of operations needed for N training points with f as dimension for feature space and N_s as total count

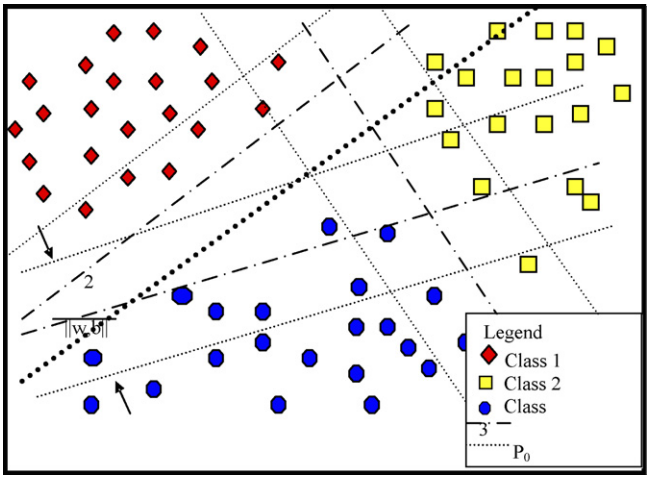


Fig. 2. Proposed multi-class SVM.

for SV, ranges from $(N_s^3 + N_s^2\, l + N_s\, f\, N)$ to $(N_s^2 + N_s\, f\, N)$ depending on the SV location with respect to the hyper planes [5,31]. Section 6 further compares performance of proposed technique with respect to standard SVM.

The optimization steps for the hyper plane parameters are detailed in the following steps. Taking partial derivatives of $L(w,b)$ with respect to the hyper plane parameters w and b for each classifier:

$$\frac{\partial L(w,b)}{\partial w_n} = 0, \quad \frac{\partial L(w,b)}{\partial b_n} = 0 \tag{12}$$

Defining:

$$a_i = 0\begin{cases} 1 & y_i = n \\ 0 & y_i \neq n \end{cases}$$

Eq. (12) becomes:

$$\begin{cases} \dfrac{w_n}{\lambda} + \displaystyle\sum_{i=1}^{N}\left[(-x_i x_i^{\mathrm{T}}(w_{y_i} - w_n) - x_i(b_{y_i} - b_n) - 2x_i)(1-a_i) + \sum_{m\neq y_i}^{c}(x_i x_i^{\mathrm{T}}(w_{y_i} - w_m) + x_i(b_{y_i} - b_m) + 2x_i)a_i\right] = 0 \\ \dfrac{b_n}{\lambda} + \displaystyle\sum_{i=1}^{N}\left[-(x_i^{T}(w_{y_i} - w_n) + (b_{y_i} - b_n) + 2)(1-a_i) + \sum_{m\neq y_i}^{c}(x_i^{\mathrm{T}}(w_{y_i} - w_m) + (b_{y_i} - b_m) + 2)a_i\right] = 0 \end{cases} \tag{13}$$

Let us define:

$$S_w = \sum_{i=1}^{N}\left[-(w_{y_i} - w_n)x_i^2(1-a_i) + \sum_{m\neq y_i}^{c}(w_{y_i} - w_m)x_i^2 a_i\right]$$
$$\Rightarrow S_w = -\sum_{i=1}^{N}(w_{y_i} - w_n)x_i^2 + \sum_{p=1}^{q(n)}x_{i_p}^2\sum_{n=1}^{c}(w_n - w_m)$$

A similar argument shows that:

$$S_b = \sum_{i=1}^{N}\left[-(b_{y_i} - b_n)x_i(1-a_i) + \sum_{m\neq y_i}^{c}(b_{y_i} - b_m)x_i a_i\right]$$
$$\Rightarrow S_b = -\sum_{i=1}^{N}(b_{y_i} - b_n)x_i + \sum_{p=1}^{q(n)}x_{i_p}\sum_{m=1}^{c}(b_n - b_m)$$

and

$$S_2 = \sum_{i=1}^{N}\left[2x_i(1-a_i) - \sum_{m\neq y_i}^{c}2x_i a_i\right]$$
$$\Rightarrow S_2 = \sum_{i=1}^{N}2x_i - \sum_{p=1}^{q(n)}2x_{i_p} - \sum_{p=1}^{q(n)}\sum_{m=1}^{c}2x_{i_p} = 2\sum_{i=1}^{N}x_i - c\sum_{p=1}^{q(n)}x_{i_p}$$

where $q(n)$ is the total number of observation belonging to a specific class and c the total number of different classes.

Applying similar reasoning for b, we can re-arrange Eq. (13) to obtain:

$$\begin{cases} \left(\dfrac{I}{\lambda} + \displaystyle\sum_{i=1}^{N}x_i x_i^{\mathrm{T}} + c\sum_{p=1}^{q(n)}x_{ip}x_{i_p}^{\mathrm{T}}\right)w_n + b_n\left(\sum_{i=1}^{N}x_i + c\sum_{p=1}^{q(n)}x_{i_p}\right) = \sum_{i=1}^{N}x_i x_i^{\mathrm{T}}w_{y_i} + \sum_{p=1}^{q(n)}x_{ip}x_{i_p}^{\mathrm{T}}\sum_{m=1}^{c}w_m + \sum_{i=1}^{N}x_i b_{y_i} + \sum_{p=1}^{q(n)}x_{i_p}\sum_{m=1}^{c}b_m + 2\sum_{i=1}^{N}x_i - 2c\sum_{p=1}^{q(n)}x_{i_p} \\ \left(\displaystyle\sum_{i=1}^{N}x_i^{\mathrm{T}} + c\sum_{p=1}^{q(n)}x_{i_p}^{\mathrm{T}}\right)w_n + b_n\left(\dfrac{1}{\lambda} + N + cq(n)\right) = \sum_{i=1}^{N}x_i^{\mathrm{T}}w_y + \sum_{p=1}^{q(n)}x_{i_p}^{\mathrm{T}}\sum_{m=1}^{c}w_m + \sum_{i=1}^{N}b_{y_i} + q(n)\sum_{m=1}^{c}b_m + 2(N-c)q(n) \end{cases} \tag{14}$$

To rewrite Eq. (14) in a matrix form, we use the series of definitions as mentioned in Table 1. This allows us to manipulate Eq. (14) and rewrite it as

$$\begin{cases} (C-G)W + (D-H)B = E \\ (D-H)^{\mathrm{T}}W + (R-Q)B = U \end{cases}$$

Table 1
Matrix notation

Matrix symbol	Matrix element
C	Diagonal matrix of size (f*c) by (f*c), the diagonal elements are composed of the square matrix c_n which is of size f $c_n = \frac{1}{\lambda} + \sum_{i=1}^{N}x_i^2 + c\sum_{p=1}^{q(n)}x_{i_p}^2$
D	Diagonal matrix of size (f*c) by c, the diagonal elements are the column vector d_n of length f: $d_n = \sum_{i=1}^{N}x_i + c\sum_{p=1}^{q(n)}x_{i_p}$
E	Column vector of size c made from $e_n = -2\sum_{i=1}^{N}x_i + 2c\sum_{p=1}^{q(n)}x_{i_p}$
H	Matrix of size (f*c) by c. The row vector is h_n of length c and of the form: $h_n = \left[\sum_{p=1}^{q(1)}x_{i_p} + \sum_{p=1}^{q(n)}x_{i_p} \quad \sum_{p=1}^{q(2)}x_{i_p} + \sum_{p=1}^{q(n)}x_{i_p} \quad \cdots \quad \sum_{p=1}^{q(c)}x_{i_p} + \sum_{p=1}^{q(n)}x_{i_p}\right]$
G	Square matrix of size (f*c) by (f*c), composed of matrix g_n of size f by c such that $g_n = \left[\sum_{p=1}^{q(1)}x_{i_p}x_{x_{i_p}}^{\mathrm{T}} + \sum_{p=1}^{q(n)}x_{i_p}x_{x_{i_p}}^{\mathrm{T}} \quad \cdots \quad \sum_{p=1}^{q(c)}x_{i_p}x_{x_{i_p}}^{\mathrm{T}} + \sum_{p=1}^{q(n)}x_{i_p}x_{x_{i_p}}^{\mathrm{T}}\right]$
Q	Square matrix of size c, made from the row vector q_n of length c $q_n = [(q(1) + q(n) \cdots (q(c) + q(n))]$
U	Column vector of size c, made from u_n $u_n = -2(N - cq(n))$
R	Square diagonal matrix of size c, the diagonal elements r_n are as follows $r_n = \frac{1}{\lambda} + N + cq(n)$

f denotes the dimensions of feature space and $q(n)$ the size of class n.

Solving these equations for W and B, we obtain

$$\begin{bmatrix} W \\ B \end{bmatrix} = \begin{bmatrix} (C-G) & (D-H) \\ (D-H)^{\mathrm{T}} & (R-Q) \end{bmatrix}^{-1} \begin{bmatrix} E \\ U \end{bmatrix} \tag{15}$$

We define matrix A to be:

$$A = \begin{bmatrix} (C-G) & (D-H) \\ (D-H)^{\mathrm{T}} & (R-Q) \end{bmatrix} \tag{16}$$

and L to be:

$$L = \begin{bmatrix} E \\ U \end{bmatrix} \tag{17}$$

These definitions enable us to rewrite Eq. (14) in a very compact form:

$$\begin{bmatrix} W \\ B \end{bmatrix} = A^{-1}L \tag{18}$$

Eq. (18) provides the separating hyper plane slopes and intercepts values for the different c classes. The hyper plane is uniquely defined based on matrices A and L and it does not depend on SV or the Lagrange multipliers.

Once the hyper plane parameters are defined, a data point is labeled according to the multi-classification decision function:

$$f(x) = \arg\max_m((w_m^{\mathrm{T}}x) + b_m), \quad m = 1, \ldots, c \tag{19}$$

4. Proposed dynamic SVM

Once the hyper plane slopes are defined incorporation of a recently acquired image sequence (x_{N+1}) into the existing model necessitates a full scale retraining for the classifier:

$$\begin{bmatrix} W \\ B \end{bmatrix}_{\mathrm{new}} = \begin{bmatrix} (C_{\mathrm{new}} - G_{\mathrm{new}}) & (D_{\mathrm{new}} - H_{\mathrm{new}}) \\ (D_{\mathrm{new}} - H_{\mathrm{new}})^{\mathrm{T}} & (R_{\mathrm{new}} - Q_{\mathrm{new}}) \end{bmatrix} \begin{bmatrix} E_{\mathrm{new}} \\ U_{\mathrm{new}} \end{bmatrix} \tag{20}$$

Clearly, this approach is computationally very expensive for applications with stream datasets. It is expensive in terms of memory requirements and computation time. To maintain an acceptable balance between storage, accuracy and computational time, we propose a dynamic SVM classifier to appropriately dispose of the recently acquired image sequences.

4.1. Dynamic strategy for sequential data

Whenever the model needs to be updated, each dynamic sequence is expected to alter matrices C, G, D, H, E, R, Q and U in Eqs. (16) and (17) by an incremental amount of $\Delta C, \Delta G, \Delta D, \Delta H, \Delta E, \Delta R, \Delta Q$ and ΔU, respectively. A classifier update is required if the confidence level in classification accuracy at the tth iteration Θ_t is smaller than the initial value Θ_{ini} (or the confusion rate CR_t is higher than the expected initial value $\mathrm{CR}_{\mathrm{ini}}$). Θ_{ini} or $\mathrm{CR}_{\mathrm{ini}}$ values can be user defined or based on pilot studies. The confidence level Θ is evaluated using elements of the confusion matrix CM. CM is a valuable visualization tool for supervised learning to easily verify if the system is mislabelling classes. The matrix columns represent the instances in a predicted class, while the matrix rows represent the instances in an actual class:

$$\mathrm{CM} = \begin{bmatrix} s_{11} & . & . & . & s_{1c} \\ s_{21} & s_{22} & . & . & s_{2c} \\ . & . & . & . & . \\ . & . & . & . & . \\ s_{c1} & . & . & . & s_{cc} \end{bmatrix}, \quad \Theta_t = \frac{\sum_{i=1}^{c} s_{ii}}{\sum_{i=1,j=1}^{c} s_{ij}} \quad \text{and} \quad \mathrm{CR} = \frac{\sum_{i=1,i\neq j}^{c} s_{ij}}{\sum_{i=1,j=1}^{c} s_{ij}}$$

where s_{ii} is the ith diagonal element in the confusion matrix and $\sum_{i=1,j=1}^{c} s_{ij}$ is the number of data belonging to class i whereas the classifier recognized them as being class j.

For illustrative purposes, let us consider a recently acquired stream data x_{N+1} belonging to class t. Eq. (20) then becomes:

$$\begin{bmatrix} W \\ B \end{bmatrix}_{\mathrm{new}} = \begin{bmatrix} (C+\Delta C) - (G+\Delta G) & (D+\Delta D) - (H+\Delta H) \\ (D+\Delta D) - (H+\Delta H) & (R+\Delta R) - (Q+\Delta Q) \end{bmatrix}^{-1} \begin{bmatrix} E+\Delta E \\ U+\Delta U \end{bmatrix}$$

To assist in the mathematical manipulation, we define the following matrices:

$$I_c = \begin{bmatrix} 1 & 0 & 0 & 0 & . & 0 \\ 0 & 1 & 0 & 0 & . & 0 \\ . & . & . & . & . & . \\ 0 & 0 & 0 & 1+c & . & 0 \\ . & . & . & . & . & . \\ 0 & 0 & 0 & 0 & . & 1 \end{bmatrix}, \quad I_t = \begin{bmatrix} 0 & 0 & . & 1 & . & 0 \\ 0 & 0 & . & 1 & . & 0 \\ . & . & . & . & . & . \\ 1 & 1 & . & 2 & . & 1 \\ . & . & . & . & . & . \\ 0 & 0 & . & 1 & . & 0 \end{bmatrix}, \quad I_e = \begin{bmatrix} 1 \\ 1 \\ . \\ 1-c \\ . \\ 1 \end{bmatrix}$$

We can then rewrite the incremental change as follows:

$$\Delta C = (x_{N+1}x_{N+1}^{\mathrm{T}})I_c; \quad \Delta G = (x_{N+1}x_{N+1}^{\mathrm{T}})I_t;$$
$$\Delta D = x_{N+1}I_c; \quad \Delta H = x_{N+1}^{\mathrm{T}}I_t;$$
$$\Delta E = -2x_{N+1}I_e; \quad \Delta R = I_c;$$
$$\Delta Q = I_t; \quad \Delta U = -2I_e.$$

The new model parameters now become:

$$\begin{bmatrix} W \\ B \end{bmatrix}_{\mathrm{ewn}} = \left[A + \begin{bmatrix} (x_{N+1}x_{N+1}^{\mathrm{T}})(I_c - I_t) & x_{N+1}^{\mathrm{T}}(I_c - I_t) \\ x_{N+1}^{\mathrm{T}}(I_c - I_t) & (I_c - I_t) \end{bmatrix}\right]^{-1} \left[L + \begin{bmatrix} -2x_{N+1}I_e \\ -2I_e \end{bmatrix}\right] \tag{21}$$

$$\text{Let : } \Delta A = \begin{bmatrix} (x_{n+1}x_{N+1}^{\mathrm{T}})(I_c - I_t) & (x_{N+1}^{\mathrm{T}})(I_c - I_t) \\ x_{N+1}^{\mathrm{T}}(I_c - I_t) & (I_c - I_t) \end{bmatrix}$$

and

$$\Delta L = \begin{bmatrix} -2x_{N+1}I_e \\ -2I_e \end{bmatrix}$$

Thus

$$\begin{bmatrix} W \\ B \end{bmatrix}_{\mathrm{ewn}} = (A + \Delta A)^{-1}(L + \Delta L) \tag{22}$$

Eq. (22) shows that the separating hyper planes slopes and intercepts for the different c classes can be efficiently and dynamically updated using the old model parameters.

4.2. Dynamic strategy for batch data

For incremental batch processing, the data is still acquired incrementally, but it is stored in a buffer in queue for chunk processing. After capturing k sequences and if the classifier needs to be updated as outlined in Section 4.1, the recently acquired data is processed and the model is updated as described by Eq. (21).

Alternately we can use the Sherman–Morrison-Woodbury [36] generalization formula described by Eq. (23) to account for the perturbation introduced by matrices M and L defined such that

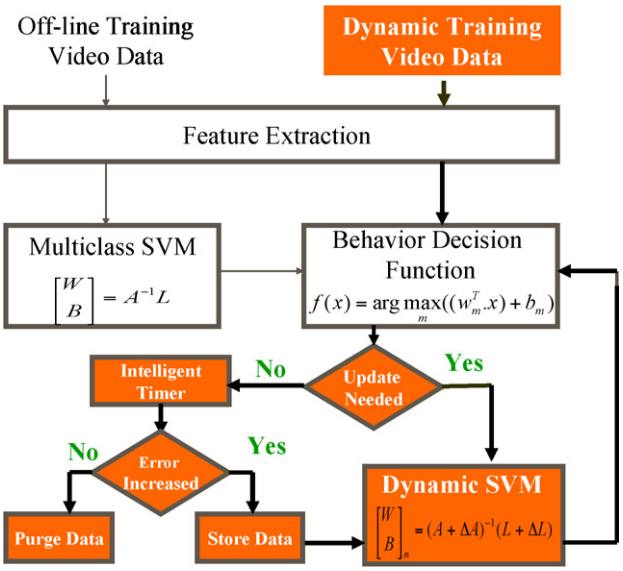


Fig. 3. Soft classification of dynamic data stream.

$(I + M^{\mathrm{T}}A^{-1}L)^{-1}$ exists:

$$(A + LM^{\mathrm{T}})^{-1} = A^{-1} - A^{-1}L(I + M^{\mathrm{T}}A^{-1}L)^{-1}M^{\mathrm{T}}A^{-1} \tag{23}$$

where

$$M = \begin{bmatrix} x_{N+1}(I_c - I_t) \\ (I_c - I_t) \end{bmatrix}; \quad L = \begin{bmatrix} x_{N+1} \\ I \end{bmatrix}^{\mathrm{T}}$$

Using Eqs. (16) and (18), the new model can represent the incrementally acquired sequences according to Eq. (24):

$$\begin{bmatrix} W \\ B \end{bmatrix}_{\text{new}} = \begin{bmatrix} W \\ B \end{bmatrix}_{\text{old}} + \begin{bmatrix} \Delta E \\ \Delta U \end{bmatrix} + \left[\begin{bmatrix} \Delta E \\ \Delta U \end{bmatrix} - \begin{bmatrix} W \\ B \end{bmatrix}_{\text{old}} \right] [I - A^{-1}M(I + M^{\mathrm{T}}A^{-1}L)^{-1}M^{\mathrm{T}}A^{-1}] \tag{24}$$

Eq. (24) shows the influence of the incremental data on calculating the new separating hyper plane slopes and intercept values for the different c classes.

Fig. 3 depicts the overall generic data flow of our proposed dynamic multi-classification algorithm.

During the initial training phase, the initial model parameters w_0 and b_0 based on matrices A and L of Eqs. (16) and (17) are stored in the cache memory. The dynamically acquired data stream is tested by the Tukey's test for outliers [36]. If the data passes the Tukey's test, the system tries to correctly predict the class label of x_{N+1} by using the decision function Eq. (19). A model update is required if at the tth iteration Θ_t is smaller than Θ_{ini}. If a model update is needed, either incremental approach described in Sections 4.1 and 4.2 are applied depending if the dynamic stream data are to be processed sequentially or in a batch manner. The recently acquired image data x_{N+1} is deleted after the model is updated. If $\Theta_t \geq \Theta_{\text{ini}}$, which means that the classifier did not require updating, the incrementally acquired images are stored in order to enable the system to learn even after several non-incremental steps. When the model is not updated, an 'intelligent timer' is incremented to keep track of the trend in the mis-classification error *Mis_Err*. If *Mis_Err* is not statistically increasing over successive non-retrain steps, the 'intelligent timer' will purge the stream video sequence stored in the buffer.

Enabling the obsolescing of non-useful data reduces the system's storage requirements. Otherwise, if *Mis_Err* is increasing, the 'intelligent timer' will get activated and force the classifier to retrain after an iteration counter defined by the user.

Table 2
Storage requirements

Classifier Type	Data structure size
Retrain_model	(1) f by c for classifier parameters (2) a permanent storage of size $(N + \text{incnum}) \times f$ that is always increasing
Dynamic_model	(1) f by c for classifier parameters (2) Temporary memory of size incnum $\times f$ for dynamic data if classifier is not updated

incnum = number of dynamic data acquired.

The addition of incremental data to the existing optimal solution still satisfies KKT conditions. Their incorporation into the initial classifier model is viewed as a 'perturbation' that insures the low probability and key data is not outnumbered. $L(w, b)$ is still convex and basically ΔA and ΔL are in equilibrium such that changes in ΔA are absorbed by changes in ΔL.

Table 2 compares the storage requirements for the proposed incremental classifier (referred to in Table 2 as Dynamic_Model) with respect to the complete retrain scenario (referred to in Table 2 as Retrain_Model). When the behavior input sequence data becomes large, the Retrain_Model storage requirements become a major concern.

5. Experimental setup

A block diagram of the experimental setup is shown in Fig. 4. It consists of a humanoid animation model that is consistent with the standards of the International Organization for Standardization (ISO) and the International Electro technical Commission (IEC) (FCD 19774) [24]. Our target is to establish interaction between actual target objects, virtual avatars and eventually retarget both virtual and actual targets. We focus on selected kinematics models to correspond to articulated target models. Using a uniquely developed graphical user interface (GUI) as described in [24], the humanoid motion is registered in the computer based on human interaction. We use selected kinematics models to enable correct behavior registration with respect to adjacency constraints and relative joint relationships. The registered behavior (typically with 50 instances per behavior) is used to train the model in an off-line mode [24]. Since several studies related to human motion classification had been published and the study of novel extraction methods and motion tracking is potentially a standalone topic [26,37], we decided to minimize the complexity of image pre-processing techniques. Further research on multi-sensor network dedicated to human tracking and identification can be found in [13,21,23,25].

However the scope of this paper is not to propose novel feature extraction techniques or motion detection. Our main objective is to demonstrate machine learning using our dynamic SVM methodology. This is why to identify motion and condense the frames into uniquely defined feature vectors we use a combination of Euclidian distances and image subtraction as proposed in our earlier work in [24]. We extract the input data by tracking color-coded marker

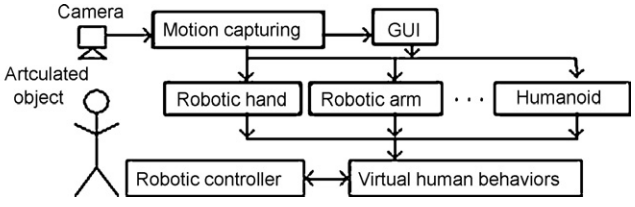


Fig. 4. Learning by visual observation modules.

Table 3
Behavioral classes for selected articulated motions

M1	Motion in right arm
M2	Motion in left arm
M3	Motion in both arms
M4	Motion in right leg
M5	Motion in left leg
M6	Motion in both legs

Table 4
Database specifics

	Feature	Class	Training	Testing
GMM	2	3	200	100
IRIS	4	3	100	50
Pentagon	2	5	66	33

points tagged to 11 joints of the humanoid. Motion detection is derived from the positional variations of the markers relative to prior frames. This extraction method results in lower storage needs without affecting the accuracy of the humanoid behavior description.

The collected raw data is a two-dimensional (2D) image sequence of the humanoid as captured by a Pulnix CCD color camera. The image sequence is treated as one unit of sensory data. For each behavior, we acquired 40 sequences each comprised of 50 frames. Every frame contained the positional information of the 11 markers. Each sequence of 50 frames is then condensed into a single vector characterizing the motion behavior of the humanoid. Since we are interested in the selected articulated motions listed in Table 3, we compute the feature vectors describing motion as the summation across the frames of the squared difference of the consecutive marker positions.

Because the limited number of training datasets is one of the inherent difficulties in the learning methodology [3] and to address this limitation, we artificially create two synthetic datasets by adding noise to the feature space. Noise source in the first dataset was modeled as a uniformly distributed noise whereas Gaussian noise with distribution ($\sigma = 1$) was incorporated to the second dataset.

6. Experimental analysis for behavior learning

We validated the proposed dynamic multi-classification method described in Sections 3 and 4 against the stream data acquired experimentally. Results are summarized in the following four subsections. Section 6.1 contrasts the off-line model with a freeware code. Section 6.2 compares the off-line and dynamic models to the retrain classifier. Section 6.3 analyses the performance of the dynamic model when updates have a sequential nature. Section 6.4 contrasts batch versus sequential model processing. Section 6.5 compares efficiency, storage and computational requirements of the proposed dynamic and retrain models. Finally Section 6.6 shows convergence rates for models that exhibit an initial poor classification performance.

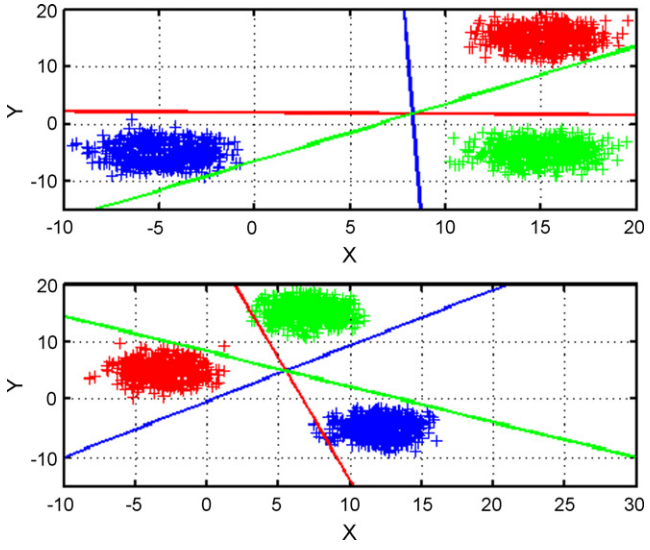


Fig. 5. Hyper planes directivity.

6.1. Comparing off-line multi-classification SVM

First, we tried our proposed multi-classification technique with different data sets to verify how the separating hyper planes will shift directivity with respect to changes in the data. Fig. 5 is a scatter plot illustrating the artificially created input datasets with the separating planes positions.

Second we compared classifier performance with the code available in the Statistical Pattern Recognition Toolbox written by Vojtech Franc and Vaclav Hlavac [27]. The freeware code is designed to solve for the Lagrange multipliers and return after a series of error optimization, the model parameters as well as the misclassification error. We used the following databases found in Matlab Stprtool Toolbox (*The MathWorks, Inc., Natick, MA*): IRIS, GMM and Pentagon. Databases specifics are listed in Table 4.We compared CR of our offline proposed approach referred to as *Prop* in Table 5 to [27]. We varied λ for optimal hyper plane positions and tried different kernel functions for [27]. Table 5 displays the experimental results. *Prop*'s performance was greatly affected by the selection of λ, whereas OAO and OAA confusion rates of [27] were heavily dependant on kernel functions. *Prop CR* was best in the Pentagon data, twice as bad for IRIS and comparable to the GMM case.

Third we compared our off-line multi-classification SVM technique on the dataset that was collected per Section 5 details. Our proposed technique resulted in a much shorter computing time than in [27] as Fig. 6 suggests. For large datasets, the code proposed in [27] did not execute because the optimization steps consumed considerable computing resources.

Fig. 6 shows *Time Ratio* computed as the execution time ratio of *Prop* over the execution time required by [27].

Due to our soft computing technique for the hyper planes, the execution time of our method is dramatically shorter than the standard optimization that solves for the Lagrange multipliers.

Table 5
Experimental results for prop vs. OAO and OAA

	λ	Kernel	GMM	Iris	Pentagon
Prop	0.005	NA	1.02%	14.58%	0%
Prop	0.5	NA	1.02%	14.58%	0%
Prop	1	NA	1.02%	12.50%	0%
Prop	10	NA	1.02%	12.50%	3.03%
Prop	100	NA	4.08%	9.50%	15.15%
Prop	1000	NA	5.10%	9.50%	21.21%
OAO	NA	RBF	1.63%	6.25%	21.70%
OAO	NA	Linear	1.40%	4.18%	11%
OAO	NA	Poly	4.08%	4.17%	2%
OAO	NA	Sigmoid	7.14%	12.92%	9.70%
OAA	NA	RBF	2%	6%	20%
OAA	NA	Linear	1.40%	4.18%	12%
OAA	NA	Poly	4.37%	4.17%	3%
OAA	NA	Sigmoid	8.10%	11.25%	9%

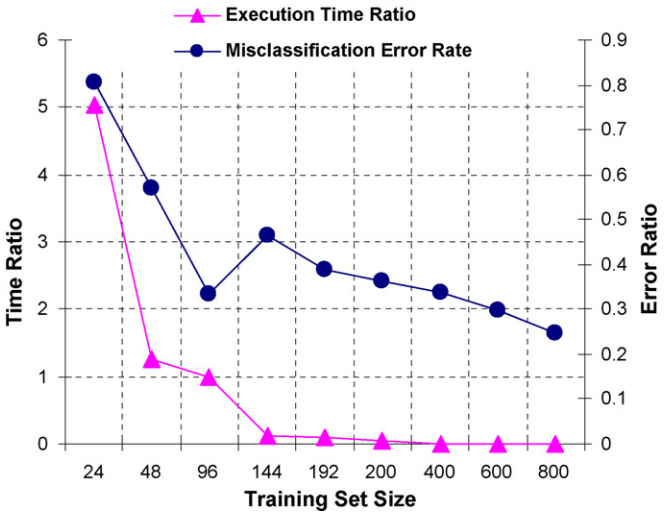


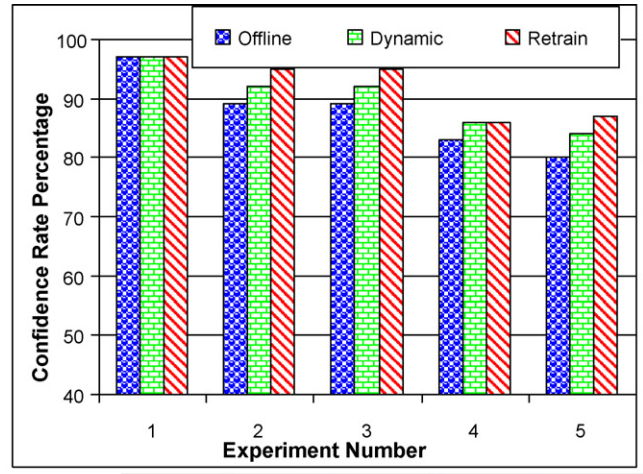
Fig. 6. *Time* and *Mis_Error* ratios.

Similarly, *Mis_Error Ratio* represents the ratio of our misclassification error divided by the error rate as calculated by [27]. As the video stream data increased in size, the misclassification error of *Prop* technique became more attractive than [27].

6.2. Analyzing articulated humanoid sequences

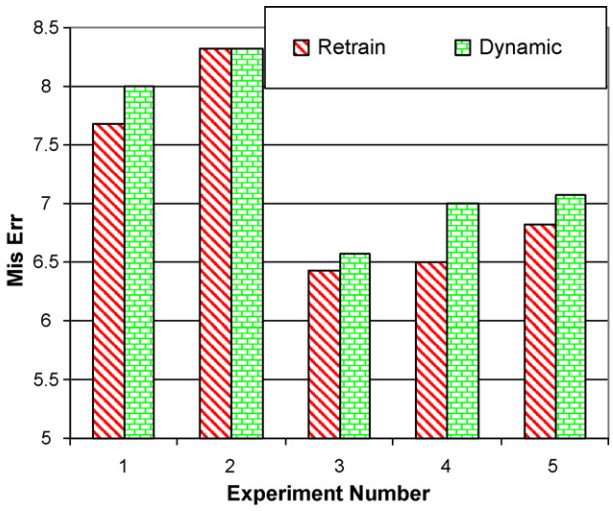
We started by investigating the effect of the data size used for training on *Mis_Err*. For this purpose, we based our analysis on a matrix of three scenarios with five different experiments for each. In all instances, we did not reuse the data sequences used for training to prevent the model from being over-fit.

- *Scenario 1*: [off-line model]: Use one training dataset to develop the off-line model. Compute the confidence rate Θ for the off-line model using a subsequent dataset.
- *Scenario 2*: [dynamic model]: Acquire and sequentially process incremental frames one at a time according to the incremental strategy highlighted in Section 4. When necessary, update the model dynamically as proposed in Section 4.1. Compute the confidence Θ rate based on the same subsequent test-set sequence used in Scenario 1.
- *Scenario 3*: [retrain model]: Acquire and incorporate incremental frames in the training set. Re-compute the model parameters. We refer to this scenario as the Retrain model. Compute the confidence rate Θ based on the same subsequent test-set sequence used in Scenarios 1 and 2.



	Experiment			Experiment			Experiment			Experiment			Experiment		
Training	30	30	35	25	25	30	20	20	30	15	15	25	10	10	20
Dynamic		5			5			10			10			10	
Testing	5	5	5	10	10	10	10	10	10	15	15	15	20	20	20

Fig. 7. Confidence rates Θ for off-line, dynamic and retrain models.



	Experiment		Experiment		Experiment		Experiment		Experiment	
Training	600	720	600	1200	600	1200	1200	1800	1200	2400
Dynamic	120		600		600		600		1200	
Testing	120	120	600	600	1200	1200	600	600	1200	1200

Fig. 8. *Mis_Err* rates for the dynamic and retrain models.

For each scenario, confidence rate Θ is shown in Fig. 7. Fig. 7 shows that the performance of the dynamic model normally lies between Scenario 1 that employs a one-time training approach and Scenario 3 that continuously retrains. Furthermore the confidence rate of Scenario 2 is within 3% of Scenario 3. The discrepancies between the confidence rates are especially noticeable in all scenarios when the training sequences are reduced in size.

6.3. Analyzing sequential synthetic data stream

We then analyzed the synthetic data using Scenarios 2 and 3 described in Section 6.2. Uniformly distributed noise was added to the feature space of the stream sequences collected during our experimental setup. Fig. 8 compares the statistical misclassification accuracy results of our proposed incremental multi-classification SVM to that of the retrain model.

In order to investigate the worst misclassified behavior classes, we computed CM shown in Table 6 for all the experiments run in Fig. 6.

Table 6
Confusion matrix for dynamic model

	M1	M2	M3	M4	M5	M6
M1	30698	0	598	0	0	0
M2	0	36926	41	60	172	0
M3	521	172	36506	0	0	0
M4	0	0	0	36831	0	368
M5	0	0	0	0	36831	368
M6	0	0	0	367	521	36311

Table 7
Confusion matrix for retrain model

	M1	M2	M3	M4	M5	M6
M1	36340	0	859	0	0	0
M2	0	36920	148	60	41	30
M3	520	82	36597	0	0	0
M4	0	30	0	36760	71	368
M5	0	19	30	160	36830	160
M6	0	11	0	881	148	36159

One observes a certain level of symmetry in the confusion occurrences in both models. This is highlighted in dark grey in Tables 6 and 7. Asymmetry in confusion is marked in light grey and is not very frequent in either model.

Based on the confusion matrices in Tables 6 and 7, we generated frequency plots to highlight the most recurring misclassification errors. Fig. 9 shows confusion rates for each model and the percentage of time when a predicted behavioral class (PC) did not match the correct behavioral class (CC).

Based on the results shown in Fig. 9, Tables 6 and 7, one can make several observations. First, the proposed dynamic SVM has fewer distinct confusion cases than the retrain model (10 versus 18 cases). Second, it has more misclassification occurrences in each confusion case. For both models, most of the confusion occurred between M1 and M3. Our proposed model has similar confusion rates when predicting class M_1 instead of M_3, and class M_3 instead of M_1.

We ran the non-parametric and distribution free Kolmogorov–Smirnov [36] comparison test to check the confidence level Θ for the dynamic and retrain models. The maximum difference between the cumulative distributions *Dist*, is 0.1856. The corresponding p value to the null hypotheses H_0 which states that the dynamic and retrain models are not different, is 0.643 and thus cannot be rejected.

We generated receiver operating characteristics (ROCs) curves for both the retrain and the dynamic models as shown in Fig. 10. Out of the four notions used to measure ROC performance, we focus on two of them: true positive (TP) and false positive (FP). TP means that the classifier correctly predicts that the new data belong to class M. FP means that the classifier predicts that the new data belong to class M when it does not actually belong to the class. Fig. 10 shows the receiver operation curve (ROC) as an outcome of the classification performance. The area under the ROC curves (AUC) for the retrain and dynamic models are respectively 0.9004 and 0.8701 which is greater than 0.5. An AUC of 0.5 or a diagonal line response represents random guessing of data stream behavior classification. Thus with AUC values close to 1, the test statistic on ROC is valid, and we can conclude that both dynamic and retrain performed well in classifying incremental data.

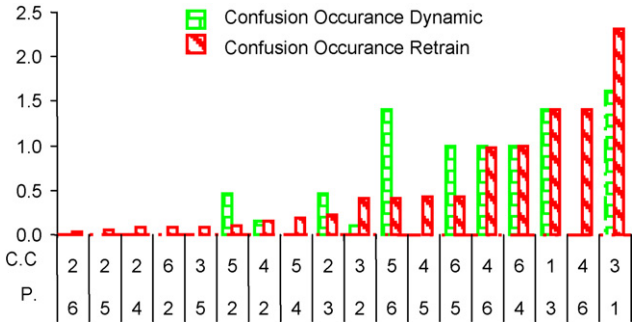


Fig. 9. Confusion occurrence for dynamic and retrain models.

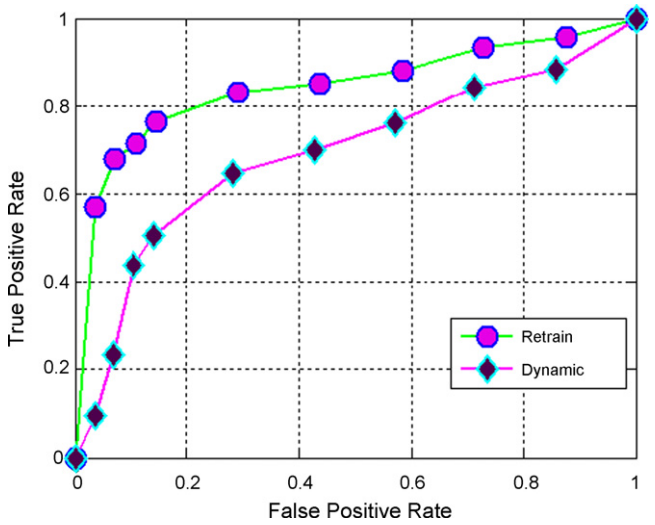
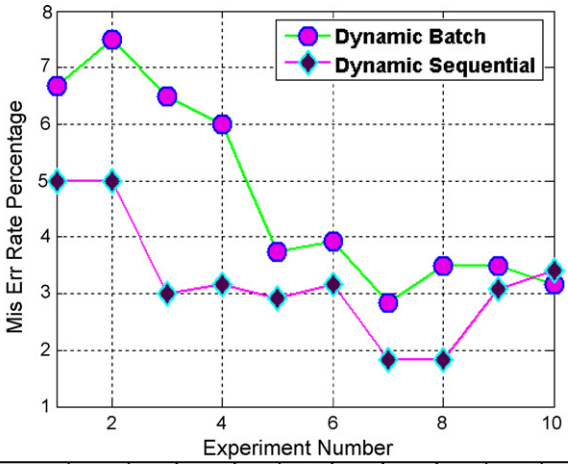


Fig. 10. Dynamic and retrain ROC curves.

6.4. Analyzing batch synthetic data stream

We compared the performance of batch to sequential processing. We added a Gaussian noise with distribution ($\sigma = 1$) to the feature space of the stream data. We processed the new datasets using our proposed incremental technique first in sequential then in batch mode (using 100 new datasets at a time).

Fig. 11 compares *Mis_Err* rates of misclassified behaviors for each mode. The performance of the two methods becomes more comparable as the training and the incremental sequence sizes are increased. Sequential processing seems to be more suited when off-line models are computed using a reduced number of training sequences because incremental data acquisition enables continuous model training in a more efficient manner than off-line training. Furthermore *Mis_Err* in Fig. 11 of the data sequences generated by adding Gaussian noise are lower than *Mis_Err* rates using data corrupted with uniformly distributed noise. Finally, with a Gaussian distributed noise, *Mis_Err* rate for the dynamic model is not statistically different from the error rate of the retrain technique.



Training set	600	720	600	1200	600	1200	1200	1800	1200	2400
Dynamic set	120		600		600		600		1200	
Testing set	120	120	600	600	1200	1200	600	600	1200	1200

Fig. 11. Batch vs. sequential processing.

Table 8
Accuracy vs. storage requirements

Proposed model *S*		Re-train model *S*	Delta
Worst case	Best case		
120*22	18*18	720*22	−0.39%
600*22	18*18	1200*22	−0.13%
1200*22	18*18	2400*22	0.08%

6.5. Storage and computational requirements versus accuracy

We compared the storage requirements *S* (referenced in Table 8), of the proposed technique to those of the retrain model considering accurate behavior classification. We considered extreme storage cases when using the proposed dynamic multi-classification procedure. The worst-case scenario (referenced as *W* in Table 8) occurred when all the incremental sequences were tested according to Section 4.1 and *Mis_Err* was less than the threshold, *Thres*. This scenario did not require a model update. However, the data had to be stored for use in future model updates to maintain the model learning ability. The best-case scenario (referred to as *B* in Table 8) occurred when *Mis_Err* for the acquired data sequences was greater than *Thres*. This scenario required temporary storage of the incremental sequence while matrix *A* was being computed for the updated model. *A* is a square matrix of size ($f^*c + c$), where *f* is equal to the dimension of features space and *c* the number of different classes.

Table 8 shows the results of this comparison. Delta is defined as an average computed across the different experiments mentioned in previous sections:

$$\text{Delta} = \frac{1}{n}\sum(\text{Dynamic\_}Mis\_Err - \text{Retrain\_}Mis\_Err)$$

From a storage perspective, the dynamic model is less demanding than the retrain model. The drawback might be a slightly higher but not significantly different *Mis_Err*. The dynamic procedure enables model tuning with an acceptable error rate for accurate behavior classification and reduced image storage requirements.

We then compared the computational requirements of the off-line, dynamic and retrain models. Fig. 12 shows the elapsed CPU execution time reported in seconds for each model.

The dynamic model is less demanding than the retrain model especially for massive datasets. It offers the advantage of soft computing and provides a good balance between accuracy and efficiency.

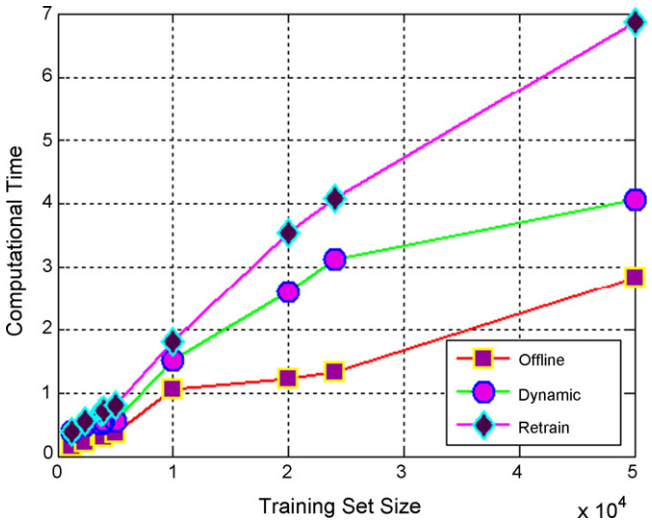


Fig. 12. Off-line, dynamic and retrain CPU time.

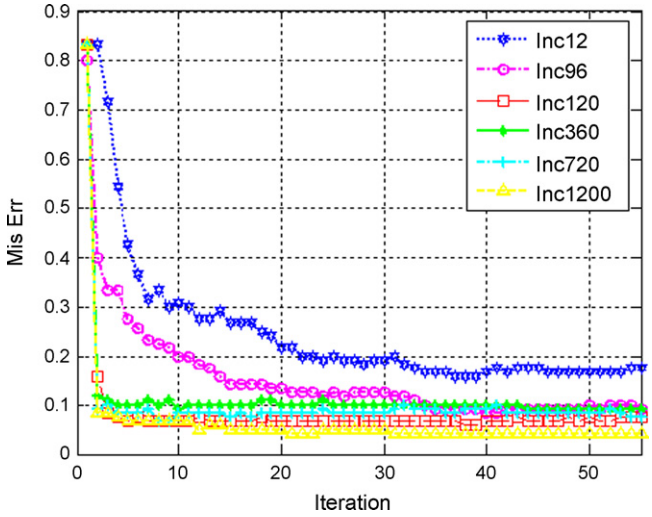


Fig. 13. Convergence rate with respect to iteration count and incremental set size.

6.6. Dynamic learning convergence rate

To explore the learning capability of the classifier when the initial model has a very poor performance as a result of a reduced training set, we ran 20 different experiments according to the workflow pictured in Fig. 3. In each experiment, we used the same initial training and testing sets. The former was composed of 12 sequences and the latter of 120. We started with an initial weak classifier model and successively incorporate dynamic data according to our proposed technique.

The incremental dataset was varied across the experiments from 12 to 1200 in order to assess the impact of the dynamic data set size on the convergence rate. Fig. 13 shows the convergence rate for selected experiments. *Mis_Err* is improved by 50% when the incremental dataset is set to be eight times bigger than the training dataset. If the dynamic dataset incorporated is set to be 100 times the training set, *Mis_Err* of the classifier drops to 10% of the initial model mis-classification rate. As it can be easily observed the incremental approach was able to gradually adjust the hyper plane positions to better classify data without the time consuming and resource extensive machine learning training phase.

7. Conclusion and future work

In this paper, we proposed a stream mining technique for articulated learning behavior by developing a unique incremental multi-classification SVM. Starting with an off-line SVM learning model, the online SVM sequentially updates the hyper plane parameters when necessary based on our proposed incremental criteria. Our classification scheme treats each image sequence as a single unit of sensory data for positional markers. The experimental results demonstrate the feasibility and merits of our technique for dynamic learning behavior using SVM. Without the need of optimization and with a convex objective function, our proposed classifier efficiently solves a single system of linear equations to find the hyper planes characteristics. It is able to describe current system activity and identify an overall motion behavior even with noisy stream data. The accuracy of the incremental SVM is comparable to a retrain model. Furthermore, our technique is attractive because it is simple to implement, it has

faster computing time and requests lower storage and memory requirements when than the retrain model.

Future work will investigate data sets involving real human motion, heterogeneous sensed data and parallel implementation for massive datasets. We will apply our proposed incremental SVM technique to benchmark data sets for behavioral learning and check for model robustness.

Acknowledgement

The authors acknowledge Jane Brooks Zurn and Xianhua Jiang for their comments. The authors would like to thank IBM Microelectronics (Essex Junction, Vermont) for the support and time used in this study. This work is partially supported by NSF Experimental Program to Stimulate Competitive Research.

References

[1] S. Amari, S. Wu, Improving support vector machines classifiers by modifying kernel functions, in: Proceedings of International Conference on Neural Networks, vol. 12, 1999, pp. 783–789.
[2] F. Aminzadeh, M. Jamshildi, Soft Computing: Fuzzy Logic Neural Networks, and Distributed Artificial Intelligence, Prentice Hall, Englewood Cliffs, 1994.
[3] M.K. Arora, P. Watanachaturaporn, SVM for classification of multi- and hyperspectral data, in: P.K. Varshney, M.K. Arora (Eds.), Advanced Image Processing Techniques for Remotely Sensed Hyperspectral Data, Springer-Verlag, 2004.
[4] L. Botton, C. Cortes, J. Denker, H. Drucker, I. Guyon, L. Jackel, Y. LeCun, U. Muller, E. Sackinger, P. Simard, V. Vapnik, Comparison of classifier methods: a case study in handwriting digit recognition, in: International Conference on Pattern Recognition, IEEE Computer Society Press, 1994, pp. 77–87.
[5] C.C. Burges, A tutorial on support vector machines for pattern recognition, in: Proceedings of the International Conference on Data Mining and Knowledge Discovery, vol. 2(2), 1998, pp. 121–167.
[6] L. Brown, J. Connell, A. Hampapur, S. Pankanti, A. Senior, Y. Tian, Smart surveillance: applications, technologies and implications, Inf. Commun. Signal Process. (2003) 1133–1138.
[7] H. Byun, S.-W. Lee, in: S.-W. Lee, A. Verri (Eds.), Applications of Support Vector Machines for Pattern Recognition: A Survey, vol. 2388, LNCS, 2002, pp. 213–236.
[8] C. Campbell, R. Herbrich, T. Graepel, Bayes point machines: estimating the Bayes point in kernel space, in: Proceedings of the IJCAI Workshop Support Vector Machines, 1999, pp. 23–27.
[9] G. Cauwenberghs, R. Genov, in: S.-W. Lee, A. Verri (Eds.), Kreneltron: Support Vector 'Machine' in Silicon, vol. 2388, LNCS, 2002, pp. 120–134.
[10] G. Cauwenberghs, T. Poggio, Incremental and decremental support vector machine learning, Adv. Neural Inf. Syst. (2000) 409–415.
[11] N. Cristianini, J. Shawe-Taylor, An Introduction to Support Vector Machines and other kernel-based learning Methods, Cambridge University Press, 2000.
[12] F. d'Alche'-Buc, L. Ralaivola, Incremental support vector machine learning: a local approach, in: Proceedings of the International Conference on Artificial Neural Networks, 2001, pp. 322–330.
[13] L.S. Davis, I. Haritaoglu, D. Harwood, W4: real-time system for detection and tracking people in 2.5d, in: Proceedings of the 5th European Conference on Computer, Vision, 1998, p. 877.
[14] R. Duda, P. Hart, D. Stock, Pattern Classification, John Willy & Sons Inc., 2001.
[15] S.I. Dworkin, J.B. Zurn, D. Hohmann, Y. Motai, A real-time rodent tracking system for both light and dark cycle behavior analysis, in: Proceedings of the IEEE Workshop on Applications of Computer Vision, 2005, pp. 87–92.
[16] R. Fletcher, Practical Methods of Optimization, John Wiley and Sons Inc., 1987.
[17] M. Flickner, I. Haritaoglu, Detection, Tracking of shopping groups in stores, in: proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, vol. 1, 2001, p. 431.
[18] D.A. Forsyth, J. Ponce, Computer Vision: A Modern Approach, Prentice Hall, 2003.
[19] F. Girosi, E. Osuna, Reducing the run time complexity of support vector machines, in: Proceedings of the International Conference on Pattern Recognition, 1998.
[20] R. Herbrich, Learning Kernel Classifiers: Theory and Algorithms, MIT Press, 2002.
[21] S. Hiura, T. Matsuyama, T. Wada, K. Murase, A. Toshioka, Dynamic memory: architecture for real time integration of visual perception, camera action, and network communication, in: Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2000, pp. 728–735.
[22] C. Hsu, C. Lin, A comparison of methods for multi-class support vector machines, IEEE Trans. Neural Netw. 13 (2002) 415–425.
[23] S. Inokuchi, A. Nakazawa, H. Kato, Human tracking using distributed vision systems, in: Proceedings of the 14th International Conference on Pattern Recognition, 1998, pp. 593–596.
[24] X. Jiang, Y. Motai, Incremental on-line PCA for automatic motion learning of eigen behavior, special issue of automatic learning and real-time, Int. J. Intelligent Syst. Technol. Appl. 2 (2007) 296–312.
[25] G. Kogut, Trivedi, I. Mikic, Distributed video networks for incident detection and management, in: Proceedings of the IEEE Conference on Intelligent Transportation Systems, 2000, pp. 390–396.
[26] B.S. Manjunah, S. Newsan, J. Testic, L. Wang, Issues in managing image and video data, in: Proceedings of SPIE International Symposium on Electronic Imaging, Storage and Retrieval Methods and Applications for Multimedia, 2004, pp. 280–292.
[27] V. Franc, V. Hlavăc, Statistical pattern recognition toolbox for Matlab, 2000–2001, http://cmp.felk.cvut.cz.
[28] S. Mukherjee, V. Vapnik, Support vector method for multivariant density estimation, in: Proceedings of the Neural Information Processing Systems, 1999, pp. 659–665.
[29] C. Nakajima, T. Poggio, M. Pontil, People recognition and pose estimations in image sequences, in: Proceedings of the IEEE International Joint Conference on Neural Networks, vol. 4, 2000, pp. 189–194.
[30] J. Platt, Fast Training of Support Vector Machines using Sequential Minimal Optimization, Advances in Kernel Methods-support Vector Learning, MIT Press, 1999, pp. 185–208.
[31] M. Pontil, A. Verri, Properties of support vector machines, AIM (1997) 1612–1617.
[32] B. Schölkopf, A.J. Smola, Learning with Kernels, MIT Press, 2002.
[33] A. Shawkat, K. Smith, On learning algorithm selection for classification, Appl. Soft Comput. 6 (2006) 119–138.
[34] J. Suykens, J. Vandewalle, Least squares support vector machine classifier, Neural Process. Lett. 9 (1999) 293–300.
[35] V.N. Vapnik, Statistical Theory, Wiley, New York, 1998.
[36] R. Witte, J. Witte, J.S. Witte, Statistics, John Wiley & Sons, 2000.
[37] S. Zelikovitz, Mining for features to improve classification, in: Proceedings of the Machine Learning, Models, Technologies and Applications, 2003, pp. 108–114.