Max-Margin Action Prediction Machine

Yu Kong, *Member, IEEE* and Yun Fu, *Senior Member, IEEE*

Abstract—The speed with which intelligent systems can react to an action depends on how soon it can be recognized. The ability to recognize ongoing actions is critical in many applications, for example, spotting criminal activity. It is challenging, since decisions have to be made based on partial videos of temporally incomplete action executions. In this paper, we propose a novel discriminative multi-scale kernelized model for predicting the action class from a partially observed video. The proposed model captures temporal dynamics of human actions by explicitly considering all the history of observed features as well as features in smaller temporal segments. A compositional kernel is proposed to hierarchically capture the relationships between partial observations as well as the temporal segments, respectively. We develop a new learning formulation, which elegantly captures the temporal evolution over time, and enforces the label consistency between segments and corresponding partial videos. We prove that the proposed learning formulation minimizes the upper bound of the empirical risk. Experimental results on four public datasets show that the proposed approach outperforms state-of-the-art action prediction methods.

Index Terms—Action prediction, action recognition, structured SVM, composite kernel, sequential data

1 Introduction

In many real-world scenarios, predicting the action of a person before it is fully executed is critical and has broad applications, such as vehicle accident avoidance, criminal activity analysis, and health care assistance. In those applications, intelligent systems do not have the luxury of waiting for the entire action execution before having to react to the action contained in it. For example, being able to predict a dangerous driving situation before it occurs; opposed to recognizing it thereafter. In addition, it is also important for an autonomous robot to stop a falling action of a person under observation as the person would most likely fall down. Unfortunately, most of existing action recognition algorithms [1], [2], [3], [4] are unsuitable for such early classification tasks as they expect to see the entire set of action dynamics extracted from a full video. Consequently, poor recognition performance can be expected if those action recognition algorithms are applied to classify partially observed action videos.

The major difference between action recognition and action prediction is that visual data arrive sequentially in action prediction, while the data are entirely observable in action recognition. Therefore, to achieve accurate prediction as early as possible, it is essential to maximize the discriminative power of the beginning temporal segments in an action video. In addition, accurate action prediction relies on effectively utilizing useful history action information. As the action data are progressively observed, the confidence of the partial history observations should also increase.

• *Y. Kong is with the Department of ECE, Northeastern University, Boston, MA. E-mail: yukong@ece.neu.edu.*
• *Y. Fu is with the Department of ECE and College of CIS, Northeastern University, Boston, MA. E-mail: yunfu@ece.neu.edu.*

Manuscript received 10 Sept. 2014; revised 18 Sept. 2015; accepted 28 Sept. 2015. Date of publication 15 Oct. 2015; date of current version 11 Aug. 2016.
Recommended for acceptance by G. Mori.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TPAMI.2015.2491928

In this paper, we propose a novel max-margin action prediction machine (MMAPM) for early recognition of unfinished actions. Our model characterizes human actions at two different temporal granularities (Fig. 1) to learn the evolution and dynamics of actions, and predicts action labels from partially observed videos containing temporally incomplete action executions. At the fine granularity, local templates are used to consider the sequential nature of human actions. The discriminative power of the beginning temporal segments are maximized by enforcing their label consistency. The temporal arrangements of these local templates also implicitly capture temporal orderings of inhomogeneous action segments.

At the coarse granularity, global templates are built to capture the history of action information. The global templates summarize action evolutions at different temporal lengths, spanning from the start of a video to the current point in time. Our model uses this information to learn how to differentiate between classes using all available information. For example, for the action class "push" the important feature is that the "arm is up", which can be used to distinguish it from the class "kick". By learning a model for increasing amount of information, our model captures the evolution of actions in each class.

Visual similarities frequently appear in action sequences. For example, a majority of subsequences in "punch" and "push" are visually similar and thus may not be linearly separable. Therefore, the linear local and global templates may not be able to differentiate various action categories. To overcome this problem, we kernelize our model, and use composite kernels to capture the complex relationships between partial observations and progress levels[1] as well as the relationships between segments and progress levels. The proposed kernel functions non-linearly map segment

1. The progress level m is the number of observed segments in a video, ranging from 1 to M: $m \in \{1, M\}$, where M is the number of segments in each full video.

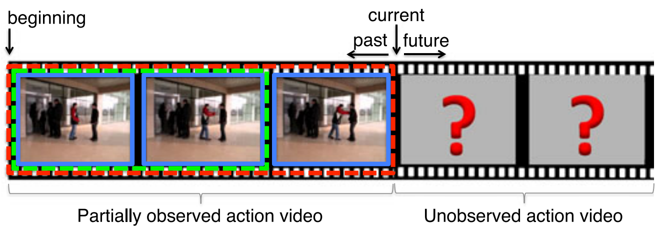


Fig. 1. Our method predicts action label given a partially observed video. Action dynamics are captured by both local templates (solid rectangles) and global templates (dashed rectangles).

and partial observation data to high-dimensional spaces, which helps us better separate visually similar data.

A new convex learning formulation based on the structured support vector machine (SSVM) is developed to consider the nature of the sequentially arriving action videos. This is achieved by introducing new constraints into the learning formulation. Equivalent dual form is also derived to incorporate the use of composite kernels. We enforce the label consistency between segments and their corresponding full video to maximize the discriminative power of the beginning temporal segments. In addition, we introduce a principled monotonic score function for the global templates. This allows us to use the prior knowledge that informative action information is increasing as the data arrive sequentially. We show in Section 3.3 that the objective of the new learning formulation minimizes an upper bound of the empirical risk of the training data.

MMAPM has numerous advantages. Firstly, MMAPM inherits the benefits of SVM, which are the convex learning formulation and the ability to deal with noisy data. Secondly, MMAPM, specifically designed for action prediction, is efficiently trained to cover all progress levels. Thirdly, MMAPM considers the partial video at two temporal scales so that the motion information of the entire partial video and the temporal segments can be well captured for prediction. Fourthly, MMAPM is a non-linear model that can better capture the classification boundaries between action categories than a linear model.

The main contribution of this paper is the development of the MMAPM, a non-linear formalism for action prediction. It uses composite kernels to accurately measure the similarity between two action sequences of different observation ratios.[2] We also show that both long-range and short-range information can be well captured in MMAPM for prediction. In addition, encouraging the discriminability of temporal segments and monotonic scoring functions for the global templates can be a rich prior knowledge for action prediction. A particular technical challenge in our formulation is the more complex learning process caused by the desire for using this prior knowledge. We use the cutting-plane algorithm [5] to address this problem. Results of extensive experiments show that the learned MMAPM outperforms state-of-the-art action prediction methods.

This paper is an extension of our previous work [6]. These extensions include: a new prediction model learning formulation with kernels; composite kernels for sequential data representation; an efficient model learning algorithm; and more experimental results on four datasets.

2. Observation ratio is defined as the ratio of the number of frames in a partially observed video to the number of frames in the full video.

1.1 Overview of Our Method

We explore the prediction of ongoing action videos and model the relationships between partial observations and progress levels. The flowchart of our approach is shown in Fig. 2. Our model consists of two components, the local progress model (LPM) and the global progress model (GPM), which capture the relationships between temporal segments and the partial observations, respectively. Considering the nature of the sequentially arriving data, we also introduce new types of constraints into the learning formulation, which elegantly capture the characteristics of increasing amount of information in the sequential data.

Training. Our method takes fully observed action videos as inputs, and models the relationships between partial observations and their progress levels. The main steps of model learning are listed here:

1) Localizing people of interest in training videos. We adopt bounding boxes or a detector [7] to extract the people from background.
2) Extracting action features. We extract features of interest points [8] and dense trajectories [9] from the bounding boxes. The bag-of-words model is applied to represent each type of features. Note that these features are extracted from both partial observations starting from the beginning of the video as well as the current observed temporal segment.
3) Learning the prediction model. The prediction model takes the action features of partial observations starting from the beginning to the current observed frame and multiple temporal segments as inputs. The model defines GPM and LPM to capture the relationships between the progress level and the global observation, as well as the progress level and the local segments, respectively (Section 3.2). The primal and dual learning formulations are shown in Sections 3.3 and 3.4, respectively. Compositional kernels for partial observations are discussed in Section 3.5. The learning algorithm is given in Section 3.6.

Testing. Given a partially observed testing video with a known progress level, we first localize the person of interest, and extract features of both interest points and dense trajectories. The action is represented using the bag-of-words model via the learned corresponding codebook. Then the action features are fed into the learned prediction model and the action label of the video can be inferred.

2 Related Work

Action Recognition: Human actions have been popularly represented by a set of quantized local spatiotemporal features, known as bag-of-words. Methods in [1], [8], [10], [11] model the appearance of local features to discriminate action classes. To improve the representation power of local features, [12], [13], [14] model the spatiotemporal structural relationships of local features, and use both the structure information and appearance information to recognize actions.

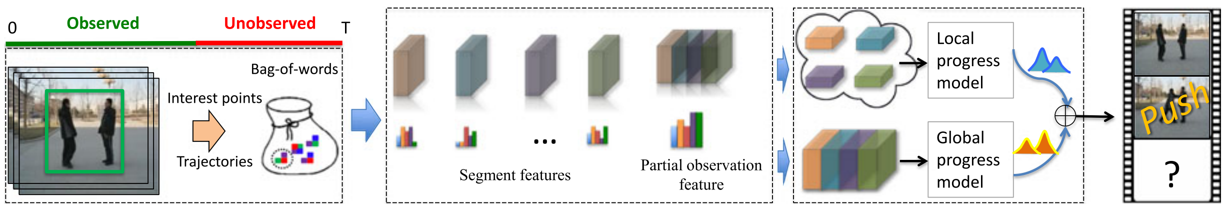


Fig. 2. Framework of our method. Given an unfinished action video, we first use bounding boxes or a detector to localize the action of interest. Interest points and dense trajectories are then extracted. Temporal segments and partial observations are both characterized using these two types of features to express both local and global information. The proposed action predictor that consists of a local progress model and a global progress model predicts the action label based on partially observed action features.

Besides local features, human actions can also be represented by trajectories of detected interest points [9], [15] and key-frames [16], [17].

Despite the success of low-level features in action recognition, they are not expressive enough to describe actions in the presence of large appearance and pose variations. This problem has been addressed by introducing human knowledge into models to describe complex human actions such as human-specified semantic descriptions [2], [3] or data-driven concepts [18]. In addition, mid-level body parts [19] can be learned from those low-level features to represent complex human motion.

Another line of work captures temporal evolutions of appearance or pose using sequential state models [20], [21], [22], [23], [24]. These approaches treat videos as a composition of temporal segments. Mid-level concepts of temporal dynamics are also captured in [22], [25], [26]. However, these methods do not model temporal action evolution with respect to progress levels. Therefore, they cannot characterize partially observed actions and are unsuitable for prediction. In contrast, we simulate sequential data arrival in prediction and use large temporal scale templates to capture action evolutions from the beginning of the video to the current observed frame. Thus, our model can recognize incomplete actions at different progress levels.

Human actions can also be recognized in other challenging scenarios. For example, actions can be learned from static images [27], [28] or arbitrary views [29], [30]. Human interactions with objects [31], [32], [33] and with human have also been investigated in [3], [34]. However, most of existing action recognition methods were designed for recognizing complete actions, assuming the action in each testing video has been fully executed. This makes these approaches unsuitable for predicting action labels in partial videos.

Action Prediction: Most of the existing work in action prediction aims at recognizing unfinished action videos. Ryoo [35] proposed the integral bag-of-words (IBoW) and dynamic bag-of-words (DBoW) approaches for action prediction. The action model of each progress level is computed by averaging features of the progress level in the same category. However, the learned model may not be representative if the action videos of the same class have large variations of appearance or pose, etc. and it is sensitive to outliers. To overcome this problem, Cao et al. [36] built action models by learning feature bases using sparse coding and used the reconstruction error in the likelihood computation. Li and Fu [37] explored long-duration action prediction problem. However, their work detects segments by motion velocity peaks, which may not be applicable on complex outdoor datasets. Lan et al. [38] built a coarse-to-fine hierarchical representation for action prediction.

An early event detector [39] was proposed to localize the starting and ending frames of an incomplete event. Activity forecasting, which aims at reasoning about the preferred path for people given a destination in a scene, has been investigated in [40]. Event prediction by learning association rules in recommendation systems was also investigated in [41], [42]. In contrast, the goal of this work is to predict action labels from partially observed videos containing temporally incomplete action executions, which is completely different from these methods.

Our work differs from existing action prediction methods [35], [36], [37], [38]. We focus on short-duration action prediction problem while [37] aimed at long-duration prediction problem. Compared with [35], [36], [37], our model incorporates an important prior knowledge that informative action information is increasing when the new observations are available. However, their methods have not taken advantage of this prior. We show in the experiments that this prior knowledge plays an important role in the prediction task. In addition, our method models label consistency of segments, which is not presented in their methods. The label consistency provides discriminative local information and implicitly captures temporal context information, which is beneficial for the prediction task. Furthermore, we capture action dynamics in both global and local temporal scales while [35], [36] captured dynamics in one single scale. Using multiple temporal scale templates allows us to capture both short-range and long-range dependencies. We also prove that our learning formulation minimizes a principled empirical risk, and provide a kernelized model in this paper.

3 Our Method

The aim of this work is to predict the action class y of a partially observed action video $x[1,t]$ before the action ends. Here 1 and t in $x[1,t]$ indicate the indices of the starting frame and the last observed frame of the partial video $x[1,t]$, respectively. Index t ranges from 1 to length T of a full video $x[1,T]$: $t \in \{1,\dots,T\}$, to generate different partial videos. An action video is usually composed of a set of inhomogeneous temporal units, which are called segments. In this work, we uniformly divide a full video $x[1,T]$ into M segments $x[\frac{T}{M}\cdot(l-1)+1, \frac{T}{M}\cdot l]$, where $l \in \{1,\dots,M\}$ is the index of a segment. The length of each segment is $\frac{T}{M}$. Note that for different videos, their lengths T may be different. Therefore, the length of

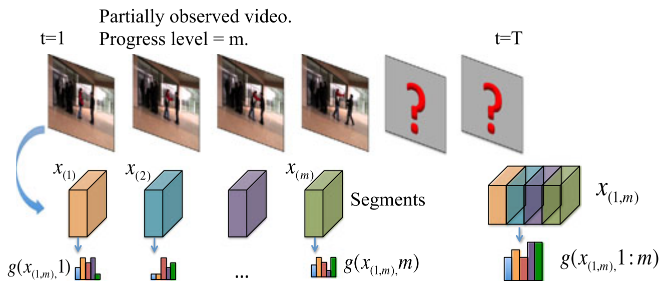


Fig. 3. Example of video segments $x_{(m)}$, partial video $x_{(1,m)}$, feature representation $g(x_{(1,m)}, l)$ of a segment ($l = 1, \dots, m$), and the representation of the partial video $g(x_{(1,m)}, 1:m)$.

segments of various videos may be different. For simplicity, let $x_{(m)}$ be the m-th segment $x[\frac{T}{M} \cdot (m-1)+1, \frac{T}{M} \cdot m]$ and $x_{(1,m)}$ be the partially observed sequence $x[1, \frac{T}{M} \cdot m]$ (see Fig. 3). The progress level m of a partially observed video is defined as the number of observed segments that the video has. The observation ratio is the ratio of the number of frames in a partially observed video $x[1,t]$ to the number of frames in the full video $x[1,T]$, which is $\frac{t}{T}$. For example, if $T = 100$, $t = 30$ and $M = 10$, then the progress level of the partially observed video $x[1,t]$ is 3 and its observation ratio is 0.3.

3.1 Action Representations

We use a bag-of-words model to represent segments and partial videos. The procedure of learning visual words for action videos is as follows. Spatiotemporal interest point (STIP) [8] and dense trajectory [9] detectors are employed to extract interest point and trajectory features from a video, respectively. The dictionaries for visual features are learned by clustering algorithms.

We denote the feature of a partial video $x_{(1,m)}$ at progress level m by $g(x_{(1,m)}, 1:m)$, which is a histogram of visual words contained in the partial video, starting from the first segment to the m-th segment (Fig. 3). The representation of the l-th ($l = 1, \dots, m$) segment $x_{(l)}$ in the partial video is denoted by $g(x_{(1,m)}, l)$, which is a histogram of visual words whose temporal locations are within the l-th segment.

3.2 Model Formulation

Let $\mathcal{D} = \{x_i, y_i\}_{i=1}^N$ be the training data, where x_i is the i-th fully observed action video and y_i is the corresponding action label. The problem of action prediction is to learn a function $f : \mathcal{X} \to \mathcal{Y}$, which maps a partially observed video $x_{(1,m)} \in \mathcal{X}$ to an action label $y \in \mathcal{Y}$ ($m \in \{1, \dots, M\}$).

We formulate the action prediction problem using the structured learning as presented in [5], [43]. Instead of searching for f, we aim at learning a discriminant function $F : \mathcal{X} \times \mathcal{Y} \to \mathcal{R}$ to score each training sample (x, y). The score measures the compatibility between a video x and an action label y. Note that, in action prediction, videos of different observation ratios from the same class should be classified as the same action category. Therefore, we use the function F to score the compatibility between the videos of different observation ratios $x_{(1,m)}$ and the action label y, where $m \in \{1, \dots, M\}$ is the progress level.

We are interested in learning a linear function $F(x_{(1,m)}, y; \mathbf{w}) = \langle \mathbf{w}, \Phi(x_{(1,m)}, y) \rangle$, which is a family of functions parameterized by $\mathbf{w}$, and $\Phi(x_{(1,m)}, y)$ is a joint feature map[3] that represents the spatio-temporal features of action y given a partial video $x_{(1,m)}$. Once the optimal model parameter $\mathbf{w}^*$ is learned, the prediction of the action label is computed by

$$\begin{aligned} y^* &= \arg\max_{y \in \mathcal{Y}} F(x_{(1,m)}, y; \mathbf{w}^*) \\ &= \arg\max_{y \in \mathcal{Y}} \langle \mathbf{w}^*, \Phi(x_{(1,m)}, y) \rangle. \end{aligned} \tag{1}$$

We define $\mathbf{w}^{\mathrm{T}} \Phi(x_{(1,m)}, y)$ as a summation of two components:

$$\begin{aligned} \mathbf{w}^{\mathrm{T}} \Phi(x_{(1,m)}, y) &= \boldsymbol{\alpha}_m^{\mathrm{T}} \psi_1(x_{(1,m)}, y) \\ &+ \sum_{l=1}^{M} \Big[\mathbf{1}(l \leqslant m) \cdot \boldsymbol{\beta}_l^{\mathrm{T}} \psi_2(x_{(1,m)}, y) \Big], \end{aligned} \tag{2}$$

where $\mathbf{w} = \{\alpha_1, \dots \alpha_M, \beta_1, \dots, \beta_M\}$ is model parameter, m is the progress level of the partial video $x_{(1,m)}$, l is the index of a segment, $\mathbf{1}(\cdot)$ is the indicator function, and $\psi_1(x_{(1,m)}, y)$ and $\psi_2(x_{(1,m)}, y)$ are feature maps for the spatio-temporal feature of action y given a partial video $x_{(1,m)}$. The two components in Eq.(2) are summarized as follows.

Global Progress Model $\boldsymbol{\alpha}_m^{\mathrm{T}} \psi_1(x_{(1,m)}, y)$ indicates how likely the action class of an unfinished action video $x_{(1,m)}$ (at progress level m) is y. It is defined as

$$\boldsymbol{\alpha}_m^{\mathrm{T}} \psi_1(x_{(1,m)}, y) = \sum_{a \in \mathcal{Y}} \alpha_m^{\mathrm{T}} \mathbf{1}(y = a) g(x_{(1,m)}, 1:m). \tag{3}$$

Here, feature vector $g(x_{(1,m)}, 1:m)$ of dimensionality D is an action representation for the partial video $x_{(1,m)}$, where features are extracted from the entire partial video, from its beginning (i.e., progress level 1) to its current progress level m. Parameter α_m of size $D \times |\mathcal{Y}|$ can be regarded as a progress level-specific template. Since the partial video is at progress level m, we select the template α_m at the same progress level, from M parameter matrices $\{\alpha_1, \dots, \alpha_M\}$. The selected template α_m is used to score the unfinished video $x_{(1,m)}$. Define $A = [\alpha_1, \dots, \alpha_M]$ as a vector of all the parameter matrices in the GPM. Then A is a vector of size $D \times M \times |\mathcal{Y}|$ encoding the weights for the configurations between progress levels and action labels, with their corresponding video evidence.

GPM simulates the sequential segment-by-segment data arrival for training action videos. Essentially, GPM captures the action appearance changes as the progress level increases, and characterizes the entire action evolution over time. In contrast to the IBoW model [35], our GPM does not assume any distributions on the data likelihood; while the IBoW model uses the Gaussian distribution. In addition, the compatibility between observation and action label in our model is given by the linear model of parameter and feature function, rather than using a Gaussian kernel function [35].

3. Feature map in this work is a feature vector defined by the two input variables.

Local Progress Model $\mathbf{1}(l \leqslant m) \cdot \boldsymbol{\beta}_l^{\mathrm{T}} \psi_2(x_{(1,m)}, y)$ indicates how likely the action classes of all the temporal segments $x_{(l)}$ $(l = 1, \ldots, m)$ in an unfinished video $x_{(1,m)}$ are all y. Here, the progress level of the partial video is m and all the segments of the video whose temporal locations l are smaller than m are considered. We define LPM as

$$\boldsymbol{\beta}_l^{\mathrm{T}} \psi_2(x_{(1,m)}, y) = \sum_{a \in \mathcal{Y}} \beta_l^{\mathrm{T}} \mathbf{1}(y = a) g(x_{(1,m)}, l), \tag{4}$$

where feature vector $g(x_{(1,m)}, l)$ of dimensionality D extracts features from the l-th segment of the unfinished video $x_{(1,m)}$. β_l of size $D \times |\mathcal{Y}|$ is the weight matrix for the l-th segment. We use the indicator function $\mathbf{1}(l \leqslant m)$ to select all the segment weight matrices, $\beta_1, \ldots, \beta_m$, whose temporal locations are smaller than or equal to the progress level m of the video. Then the selected weight matrices are used to score the corresponding segments. Let $B = [\beta_1, \ldots, \beta_M]$ be a vector of all the parameters in the LPM. Then B is a vector of size $D \times M \times |\mathcal{Y}|$ encoding the weights for the configurations between segments and action labels, with their corresponding segment evidence.

LPM considers the sequential nature of a video. The model decomposes a video of progress level m into segments, and describes temporal dynamics of segments. Note that action data preserve temporal relationships between segments. Therefore, the discriminative power of segment $x_{(m)}$ is critical to the prediction of $x_{(1,m)}$ given the prediction results of $x_{(1,m-1)}$. In this work, the segment score $\beta_m^{\mathrm{T}} g(x_{(1,m)}, m)$ measures the compatibility between the segment $x_{(m)}$ and all the classes. To maximize the discriminability of the segment, the score difference between the ground-truth class and all the other classes is maximized in our learning formulation. Thus, accurate prediction can be achieved using the newly-introduced discriminative information in the segment $x_{(m)}$.

3.3 Structured Learning Formulation

The MMAPM is formulated based on the structured SVM [5], [43]. The optimal model parameter $\mathbf{w}^*$ of MMAPM in Eq. (1) is learned by solving the following convex problem given training data $\{x_i, y_i\}_{i=1}^{N}$:

$$\min \frac{1}{2}\|\mathbf{w}\|^2 + C(\xi_1 + \xi_2 + \xi_3) \tag{5}$$

$$\begin{aligned} &\text{s.t. } \forall m, \forall(\bar{y}_1, \ldots, \bar{y}_N) \in \mathcal{Y}^N, \\ &\frac{1}{N}\sum_{i=1}^{N}[\mathbf{w}^{\mathrm{T}}\Phi(x_{i(1,m)}, y_i) - \mathbf{w}^{\mathrm{T}}\Phi(x_{i(1,m)}, \bar{y}_i)] \\ &\geqslant \frac{M}{N}\sum_{i=1}^{N}\delta(\bar{y}_i, y_i) - \frac{\xi_1}{u(m/M)}, \end{aligned} \tag{6}$$

$$\begin{aligned} &m = 2, \ldots, M, \forall(\bar{y}_1, \ldots, \bar{y}_N) \in \mathcal{Y}^N, \\ &\frac{1}{N}\sum_{i=1}^{N}[\boldsymbol{\alpha}_m^{\mathrm{T}}\psi_1(x_{i(1,m)}, y_i) - \boldsymbol{\alpha}_{m-1}^{\mathrm{T}}\psi_1(x_{i(1,m-1)}, \bar{y}_i)] \\ &\geqslant \frac{M}{N}\sum_{i=1}^{N}\delta(\bar{y}_i, y_i) - \frac{\xi_2}{u(m/M)}, \end{aligned} \tag{7}$$

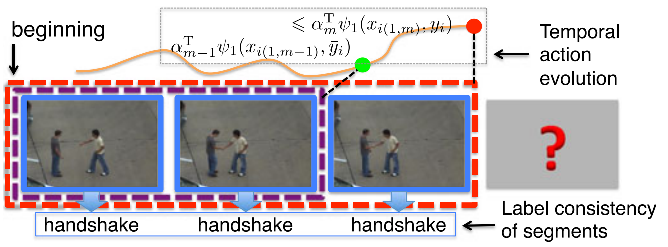

beginning
$\leqslant \alpha_m^{\mathrm{T}}\psi_1(x_{i(1,m)}, y_i)$
$\alpha_{m-1}^{\mathrm{T}}\psi_1(x_{i(1,m-1)}, \bar{y}_i)$
Temporal action evolution
?
handshake
handshake
handshake
Label consistency of segments

Fig. 4. Graphical illustration of the temporal action evolution over time and the label consistency of segments. Blue solid rectangles are LPMs, and purple and red dashed rectangles are GPMs.

$$\begin{aligned} &\forall m, \forall(\bar{y}_1, \cdots, \bar{y}_N) \in \mathcal{Y}^N, \\ &\frac{1}{N}\sum_{i=1}^{N}[\boldsymbol{\beta}_m^{\mathrm{T}}\psi_2(x_{i(m)}, y_i) - \boldsymbol{\beta}_m^{\mathrm{T}}\psi_2(x_{i(m)}, \bar{y}_i)] \\ &\geqslant \frac{M}{N}\sum_{i=1}^{N}\delta(\bar{y}_i, y_i) - \frac{\xi_3}{u(m/M)}, \end{aligned} \tag{8}$$

where C is the slack trade-off parameter similar to that in the SVM. ξ_1, ξ_2 and ξ_3 are slack variables. $u(\cdot)$ is a scaling factor function: $u(p) = p$. $\delta(\bar{y}_i, y_i)$ is the $0-1$ loss function. We use the so-called 1-slack formulation proposed in [5], which is proved to be equivalent to the n-slack formulation but is more efficient due to the smaller dual problem.

The slack variables ξ_1 and the Constraint (6) are the usual 1-slack SVM constraints [5] on the class labels. We enforce this constraint for all the progress levels m since we are interested in learning a classifier that can correctly recognize partially observed videos with different progress levels m. Therefore, we simulate the segment-by-segment data arrival for training and augment the training data with partial videos of different progress levels. The loss function $\delta(\bar{y}_i, y_i)$ measures the recognition error of a partial video and the scaling factor $u(\frac{m}{M})$ scales the loss based on the length of the partial video.

Constraint (7) considers **temporal action evolution** over time. We assume that the score $\boldsymbol{\alpha}_m^{\mathrm{T}}\psi_1(x_{i(1,m)}, y_i)$ of the partial observation $x_{i(1,m)}$ at progress level m and the ground truth label y_i must be greater than the score $\boldsymbol{\alpha}_{m-1}^{\mathrm{T}}\psi_1(x_{i(1,m-1)}, \bar{y}_i)$ of a previous observation $x_{i(1,m-1)}$ at progress level $m-1$ and all incorrect labels y (Fig. 4). This provides a monotonically increasing score function for partial observations and elaborately characterizes the nature of sequentially arriving action data in action prediction. The slack variable ξ_2 allows us to model outliers.

The slack variables ξ_3 and the Constraint (8) are used to maximize the discriminability of segments $x_{(m)}$. We encourage the **label consistency** between segments and the corresponding full video due to the nature of sequential data in action prediction (Fig. 4). Assume a partial video $x_{(1,m-1)}$ has been correctly recognized, then the segment $x_{(m)}$ is the only newly-introduced information and its discriminative power is the key to recognizing the video $x_{(1,m)}$. Furthermore, context information of segments is implicitly captured by enforcing the label consistency. It is possible that some segments from different classes are visually similar and may not be linearly separable. We use the slack variable ξ_3 for each video to allow some segments of a video to be treated as outliers.

Empirical Risk Minimization: We define $\Delta(\bar{y}_i, y_i)$ as the function that quantifies the loss for a prediction $\bar{y}_i$, if the ground-truth is y_i. Therefore, the loss of a classifier $f(\cdot)$ for action prediction on a video-label pair (x_i, y_i) can be quantified as $\Delta(f(x_i), y_i)$. Usually, the performance of $f(\cdot)$ is given by the empirical risk $R_{\text{emp}}(f) = \frac{1}{N}\sum_{i=1}^{N}\Delta(f(x_i), y_i)$ on the training data (x_i, y_i), assuming data samples are generated i.i.d.

The nature of continual evaluation in action prediction requires aggregating the values of loss quantities computed during the action sequence process. Define the loss associated with a prediction $\bar{y}_i = f(x_{i(1,m)})$ for an action x_i at progress level m as $\Delta(\bar{y}_i, y_i)u(\frac{m}{M})$. Here $\Delta(\bar{y}_i, y_i)$ denotes the misclassification error, and $u(\frac{m}{M})$ is the scaling factor that depends on how many segments have been observed. In this work, we use summation to aggregate the loss quantities. This leads to an empirical risk for N training samples: $R_{\text{emp}}(f) = \frac{1}{N}\sum_{i=1}^{N}\sum_{m=1}^{M}\{\Delta(\bar{y}_i, y_i)u(\frac{m}{M})\}$.

Denote by $\boldsymbol{\xi}_1^*$, $\boldsymbol{\xi}_2^*$ and $\boldsymbol{\xi}_3^*$ the optimal solutions of the slack variables in Eq. (5-8) for a given classifier f, we can prove that $\frac{1}{N}\sum_{i=1}^{N}(\xi_1^* + \xi_2^* + \xi_3^*)$ is an upper bound on the empirical risk $R_{\text{emp}}(f)$ and the learning formulation given in Eq. (5-8) minimizes the upper bound of the empirical risk $R_{\text{emp}}(f)$. Please refer to Appendix for details.

3.4 Kernalized Action Predictor

We first derive an equivalent and compact optimization problem for the optimization problem (5-8). Then the dual form of the new optimization problem can be easily derived and solved using the existing structured SVM solver [5].

The optimization problem (5-8) can be equivalently rewritten as

$$\min \frac{1}{2}\|\mathbf{w}\|^2 + C\xi \tag{9}$$

$$\begin{aligned}
\text{s.t. } & \frac{1}{NM}\sum_{i=1}^{N}\sum_{m=1}^{M}\{\mathbf{w}^{\text{T}}[\Phi(x_{i(1,m)}, y_{im}) + U(x_{i(1,m)}, y_{im}) \\
& + V(x_{i(m)}, y_{im})] - \mathbf{w}^{\text{T}}[\Phi(x_{i(1,m)}, \bar{y}_{im}) \\
& + U(x_{i(1,m-1)}, \bar{y}_{i(m-1)}) + V(x_{i(m)}, \bar{y}_{im})]\} \\
& \geqslant \frac{1}{N}\sum_{i=1}^{N}\sum_{m=1}^{M}\Delta(\bar{y}_{im}, y_{im}) - \frac{\xi}{u(m/M)}, \\
& \forall(\bar{y}_{11}, \ldots, \bar{y}_{NM}) \in \mathcal{Y}^{NM},
\end{aligned} \tag{10}$$

where $\xi = \xi_1 + \xi_2 + \xi_3$, and $\Delta(\bar{y}_{im}, y_{im}) = 3\delta(\bar{y}_{im}, y_{im})$. U and V are auxiliary feature maps that satisfy $\mathbf{w}^{\text{T}}U(x_{i(1,m)}, y) = \boldsymbol{\alpha}_m^{\text{T}}\psi_1(x_{i(1,m)}, y)$ and $\mathbf{w}^{\text{T}}V(x_{i(m)}, y) = \boldsymbol{\beta}_m^{\text{T}}\psi_2(x_{i(m)}, y)$, respectively. The two feature maps, $U(x_{i(1,m)}, y)$ and $V(x_{i(m)}, y)$, represent the spatio-temporal features of an action y given observations $x_{i(1,m)}$ and $x_{i(m)}$, respectively. y_{im} is the label for a partial observation $x_{i(1,m)}$ and $y_{im} = y_i$.

The model defined in Eq. (2) is a linear model for action prediction. However, in sequentially arriving action data, there may be visually similar temporal segments appearing in different action categories, e.g., "punch" and "push". Consequently, linear models may confuse those temporal segments, and the recognition performance would be degraded. To solve this problem, we propose a kernelized prediction model, a non-linear model that can learn complex non-linear classification boundaries for differentiating visually similar segments.

To introduce kernels to MMAPM, we derive the following dual form of the optimization problem (9-10):

$$\max \frac{1}{N}\sum_{\bar{\mathbf{y}}im}\pi_{\bar{\mathbf{y}}}\Delta(\bar{y}_{im}, y_{im}) - \frac{1}{2}\sum_{\bar{\mathbf{y}}}\sum_{\bar{\mathbf{y}}'}\pi_{\bar{\mathbf{y}}}\pi_{\bar{\mathbf{y}}'}H(\bar{\mathbf{y}}, \bar{\mathbf{y}}') \tag{11}$$

$$\text{s.t. } \sum_{\bar{\mathbf{y}}}\frac{\pi_{\bar{\mathbf{y}}}}{u(m/M)} = C, \tag{12}$$

where

$$\begin{aligned}
H(\bar{\mathbf{y}}, \bar{\mathbf{y}}') = & \frac{1}{N^2M^2}\left\{\sum_{i=1}^{N}\sum_{m=1}^{M}[\Psi_{im}^{+} - \Psi_{im}^{-}]\right\}^{\text{T}} \\
& \cdot\left\{\sum_{i=1}^{N}\sum_{m=1}^{M}[\Psi_{im}^{+} - \Psi_{im}^{-}]\right\},
\end{aligned} \tag{13}$$

and

$$\begin{aligned}
\Psi_{im}^{+} &= \Phi(x_{i(1,m)}, y_{im}) + U(x_{i(1,m)}, y_{im}) \\
&\quad + V(x_{i(m)}, y_{im}), \\
\Psi_{im}^{-} &= \Phi(x_{i(1,m)}, \bar{y}_{im}) + U(x_{i(1,m-1)}, \bar{y}_{i(m-1)}) \\
&\quad + V(x_{i(m)}, \bar{y}_{im}).
\end{aligned} \tag{14}$$

Note that $U(x_{i(1,m-1)}, \bar{y}_{i(m-1)}) = 0$ if $m = 1$. Here, $\pi_{\bar{\mathbf{y}}}$ is the Lagrangian multiplier. $H(\cdot)$ in the above formulation can be computed by kernel functions, which implicitly map data in the feature space into kernel space. In this work, we define composite kernels that map temporal segments and partial observations to the kernel space. This allows us to measure the similarity between two partial observations with different observation ratios.

3.5 Composite Kernels

Recent work [44], [45], [46], [47], [48] shows that kernelized models typically achieve superior performance over linear models due to the ability of learning non-linear classification boundaries. In this work, MMAPM is kernelized to further improve its prediction performance. We use kernel functions K to compute the four inner products $\Psi_{im}^{+}\Psi_{i'm'}^{+}$, $\Psi_{im}^{+}\Psi_{i'm'}^{-}$, $\Psi_{im}^{-}\Psi_{i'm'}^{+}$, and $\Psi_{im}^{-}\Psi_{i'm'}^{-}$ in Eq. (13) explicitly. In the following, we show how to compute the kernel for $\Psi_{im}^{+}\Psi_{i'm'}^{-}$, i.e. $K(\Psi_{im}^{+}, \Psi_{i'm'}^{-})$. Similar techniques can be applied for computing kernels $K(\Psi_{im}^{+}, \Psi_{i'm'}^{+})$, $K(\Psi_{im}^{-}, \Psi_{i'm'}^{+})$, and $K(\Psi_{im}^{-}, \Psi_{i'm'}^{-})$ for $\Psi_{im}^{+}\Psi_{i'm'}^{+}$, $\Psi_{im}^{-}\Psi_{i'm'}^{+}$, and $\Psi_{im}^{-}\Psi_{i'm'}^{-}$, respectively.

Consider that a partial observation consists of multiple temporal segments, each of which represents human motion in a short time frame. If the similarity between two observations is only measured using their global features, it ignores the similarity in temporal orderings as well as the similarity in appearance in a short period. For example, a "standing up" action can be mistakenly predicted as "sitting down" if we do not take appearance of temporal segments and segment orderings into account.

To solve this problem, we define our composite kernel K to measure the similarity between two partial observations

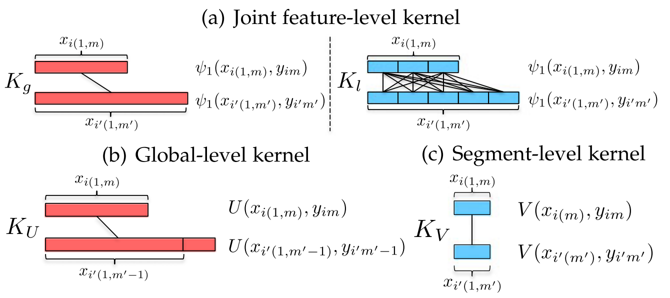

(a) Joint feature-level kernel
K_g
$x_{i(1,m)}$
$\psi_1(x_{i(1,m)}, y_{im})$
$\psi_1(x_{i'(1,m')}, y_{i'm'})$
$x_{i'(1,m')}$
K_l
$\psi_1(x_{i(1,m)}, y_{im})$
$\psi_1(x_{i'(1,m')}, y_{i'm'})$
(b) Global-level kernel
K_U
$U(x_{i(1,m)}, y_{im})$
$U(x_{i'(1,m'-1)}, y_{i'm'-1})$
$x_{i'(1,m'-1)}$
(c) Segment-level kernel
K_V
$V(x_{i(m)}, y_{im})$
$V(x_{i'(m')}, y_{i'm'})$

Fig. 5. Illustration of the components in the composite kernel $K(\Psi^+_{im}, \Psi^-_{i'm'})$. Red rectangles are partial observations and the blue ones are temporal segments.

with different observation ratios based on their global representations, local segment representations, and implicit temporal orderings. This composite kernel is defined as the summation of the kernels over all the feature functions, including Φ, U, and V defined in Eq. (9)-(10):

$$\begin{aligned}K(\Psi^+_{im}, \Psi^-_{i'm'}) = K_\Phi(\Phi(x_{i(1,m)}, y_{im}), \Phi(x_{i'(1,m')}, \bar{y}_{i'm'})) \\ + K_U(U(x_{i(1,m)}, y_{im}), U(x_{i'(1,m'-1)}, \bar{y}_{i'm'-1})) \\ + K_V(V(x_{i(m)}, y_{im}), V(x_{i'(m')}, \bar{y}_{i'm'})).\end{aligned} \tag{15}$$

The three kernel functions, K_Φ,K_U, and K_V, are discussed as follows.

Joint feature-level kernel K_Φ (Fig. 5a) computes the similarity between two feature maps of corresponding partial observations. The kernel is defined by summarizing information from GPM and LPM:

$$\begin{aligned}K_\Phi = K_g\big(\psi_1(x_{i(1,m)}, y_{im}), \psi_1(x_{i'(1,m')}, \bar{y}_{i'm'})\big) \\ + K_l\big(\psi_2(x_{i(1,m)}, y_{im}), \psi_2(x_{i'(1,m')}, \bar{y}_{i'm'})\big).\end{aligned} \tag{16}$$

The joint feature-level kernel is essentially a composite kernel that computes the similarity based on the global observation $\psi_1(\cdot,\cdot)$ starting from the beginning to the current progress level, and all the m local segments $\psi_2(\cdot,\cdot)$. Note that $K_\Phi(\cdot,\cdot)$ can measure the similarity between two partial observations with different progress levels.

Global-level kernel K_U (Fig. 5b) computes the appearance similarity between the feature map of a partial observation at progress level m and the feature map of the other partial observation at progress level $m'-1$. If $m = m'$, then K_U computes the similarity between an observation and it's previous observation.

We define K_U as a summation of kernels of various feature types:

$$K_U = \sum_{j=1}^{J} K_U^{(j)}(U(x_{i(1,m)}, y_{im}), U(x_{i'(1,m'-1)}, \bar{y}_{i'm'-1})), \tag{17}$$

where J is the number of feature types and $K_U^{(j)}$ is the kernel based on the j-th visual feature.

Segment-level kernel K_V (Fig. 5c) computes the similarity between two feature maps of two segments, i.e., $V(x_{i(m)}, y_{im})$ and $V(x_{i'(m')}, \bar{y}_{i'm'})$. Segment-level kernel K_V also implicitly captures the similarity in temporal orderings as it compares segments at different temporal locations.

We also define K_V as a summation of kernels of different feature types:

$$K_V = \sum_{j=1}^{J} K_V^{(j)}(V(x_{i(m)}, y_{im}), V(x_{i'(m')}, \bar{y}_{i'm'})), \tag{18}$$

where J is the number of feature types and $K_V^{(j)}$ is the kernel based on the j-th visual feature.

Algorithm 1. Cutting Plane Model Learning Algorithm

```
 1: Input: ${(x_i^t, y_i^t)}_{i=1}^{n_t} \subset \mathbb{R}^d \times \mathbb{R} (t = 1, \dots, m)$
 2: Output: $\mathbf{w}$
 3: Initialize working set $\mathcal{W}$
 4: repeat
 5:   Solve Eq. (11-12) using current working set $\mathcal{W}$.
 6:   for $i = 1$ to $n$ do
 7:     for $m = 1$ to $M$ do
 8:       Compute $\bar{y}_{im}$ for the $i$-th sample by Eq. (19).
 9:     end for
10:   end for
11:   $\mathcal{W} \leftarrow (\bar{y}_{11}, \dots, \bar{y}_{NM})$.
12: until No $\mathcal{W}$ changes.
```

3.6 Model Learning and Testing

Learning. The cutting plane learning algorithm [5] is applied in this work to efficiently find the optimal solution. Algorithm 1 provides a summary of the proposed training algorithm. Cutting plane algorithm iteratively finds the most violated constraints, and adds them to the working set. The optimization problem is solved on a subset of constraints in the working set. The algorithm stops when the working set does not change between two iterations.

One of the key steps in the cutting plane training algorithm is finding the most violated constraints or the separation oracle. This step solves an augmented inference step and finds the optimal label for one sample. Different from conventional action recognition tasks, action data arrive sequentially in action prediction task. Consequently, information of various temporal segments in one action sample needs to be summarized. In this work, the augmented inference problem for the partial observation in the i-th training sample is solved by

$$y^*_{im} = \arg\max_y u\Big(\frac{m}{M}\Big)\Big[\frac{1}{N}\Delta(\bar{y}_{im}, y_i) + \frac{1}{NM}\mathbf{w}^{\mathrm{T}}\Psi^-_{im}\Big]. \tag{19}$$

The model parameter $\mathbf{w}$ used here can be reconstructed from the learned support vectors and coefficients. Given a partial observation at progress level m, our inference method summarizes all the observed temporal segments in one action video in Ψ^-_{im}. A scaling function $u(\frac{m}{M})$ is used here to weigh the importance of the current partial observation. The augmented inference problem here finds the optimal y^*_{im} for all the three types of constraints, and adds it to the active working set. An alternative solution is to find y^* for each type of constraints, i.e., three violations for each video at progress level m. However, this strategy will enlarge the QP problem size, and thus results in a slow training speed as it adds more violations to the working set. Note that the inferred labels for partial videos at differegress levels in one action sample could be different.

Testing. Given an unfinished action video with progress level m (m is known in testing), our goal is to infer the class label y^* using the learned model parameter $\mathbf{w}^*$: $y^* = \operatorname{argmax}_{y \in \mathcal{Y}} \langle \mathbf{w}^*, \Phi(x_{(1,m)}, y) \rangle$. Note that during testing phase, it does not require sophisticated inference algorithms such as belief propagation or graph cut since we do not explicitly capture segment interactions. Instead, the context information between segments is implicitly captured in our model by the label consistency in Constraint (8).

3.7 Model Properties

We highlight here key properties of our model, and show some differences from existing methods.

Multiple temporal scales. Our method captures action dynamics in multiple temporal scales, while [35], [36], [38], [39] only used a single temporal scale. The use of multiple temporal scales helps us capture both short-range and long-range dependencies.

Temporal evolution over time. Our work uses the prior knowledge of temporal action evolution over time. Inspired by [39], we introduce a principled monotonic score function for the GPM to capture this prior knowledge. However, [39] aimed at finding the starting frame of an event while our goal is to predict action class of an unfinished video. The methods in [35], [36], [37], [38] do not use this prior.

Segment label consistency. We effectively utilize the discriminative power of local temporal segments by enforcing label consistency of segments. However, [35], [36], [37], [38], [39] do not consider the label consistency. The consistency also implicitly models temporal segment context by enforcing the same label for segments while [35], [36], [39] explicitly treat successive temporal segments independently.

Principled empirical risk minimization. We propose a principled empirical risk minimization formulation in this work, while it is not discussed in [35], [36], [37], [38].

Kernelized model. MMAPM uses composite kernels to implicitly map partial observations to a feature space so that visual similarity in temporal segments can be correctly classified. However, the kernel scheme is not used in [35], [36], [37], [38], [39].

4 Experiments

4.1 Datasets and Experiment Setups

We test the proposed MMAPM approach on four datasets: the UT-Interaction dataset (UTI) Set 1 (UTI #1) and Set 2 (UTI #2) [49], the BIT-Interaction dataset (BIT) [50], and the UCF11 dataset [51]. The UTI #1 dataset was captured on a parking lot with mostly static with little camera jitter. The UTI #2 dataset was taken on a lawn on a windy day with slightly background dynamics (e.g. tree moves) and camera jitters. Both of the two sets consist of 6 types of human actions, with 10 videos per class. We conduct the leave-one-out training scheme on these two datasets. The BIT dataset consists of eight types of human actions between two people, with 50 videos per class. For this dataset, a random sample of 272 videos is chosen as training samples to train MMAPM, and the remaining 128 videos are used for testing. Spatiotemporal interest points and dense trajectories are extracted from these three datasets. The dictionary size for interest point descriptors and dense trajectory features are both set to 500. The UCF11 dataset is an updated version of the Youtube action dataset used in [51]. UCF11 dataset is very challenging due to large variations in camera motion, object appearance and pose, object scale, viewpoint, cluttered background, illumination conditions, etc. It contains 1,597 videos in 11 action categories, bigger than the Youtube action dataset that contains 1,168 videos. The first 15 groups of videos are chosen as training data, three groups of videos are used for cross-validation, and the remaining seven groups are used as testing data. Spatiotemporal interest points are extracted from videos in the UCF11 dataset, and they are encoded by the bag-of-words model with a dictionary of size 500. Chi-square kernels are applied for all the composite kernels K_Φ, K_U, and K_V.

MMAPM is evaluated for classifying videos of incomplete action executions using 10 observation ratios, from 0.1 to 1, representing the increasing amount of sequential data with time. For example, if a full video containing T frames is used for testing at the observation ratio of 0.3, the accuracy of MMAPM is evaluated by presenting it with the first $0.3 \times T$ frames. At observation ratio of 1, the entire video is used, at which point MMAPM acts as a conventional action recognition model. Unless specified, the progress levels m of testing videos are known to all methods in our experiments.

4.2 Results on UTI $\#1$ and UTI $\#2$ Dataset

Action Prediction. MMAPM is compared with DBoW and IBoW [35], MMED [39], MSSC and SC [36], the method in [16], and hierarchical movemes (HM) [38]. KNN-nonDynamic, KNN-Dynamic, and the baseline method implemented in [36] are also used in comparison. The same experiment settings in [36] are followed in our experiments.

Fig. 6a shows the prediction results of all the comparison methods on UTI #1 dataset. Our MMAPM achieves better overall performance over all the other comparison approaches. Our method outperforms MSSC because we not only model segment dynamics but also characterize temporal evolutions of actions. Our method achieves an impressive 78.33 percent recognition accuracy when only the first 50 percent frames of testing videos are observed. This result is even higher than the SC method with full observations. Results of MMAPM are significantly higher than the DBoW and IBoW for all observation ratios. This is mainly due to the fact that the action models in our work are discriminatively learned while the action models in the DBoW and IBoW are computed by averaging feature vectors in a particular class. Therefore, the action models in the DBoW and IBoW may not be the representative models and are sensitive to outliers. MMED does not perform well as other prediction approaches since it is optimized for early detection of the starting and ending frame of an action. This is a different goal from this paper, which is to classify unfinished actions. MMAPM achieves superior results over the HM [38] in six out of 10 cases, and obtains the same results in two cases. MMAPM also outperforms MTSSVM due to the use of non-linear composite kernels. Action recognition results on full observations are summarized in Table 2.

Comparison results on the UTI #2 dataset are shown in Fig. 6b. MMAPM achieves better performance over all

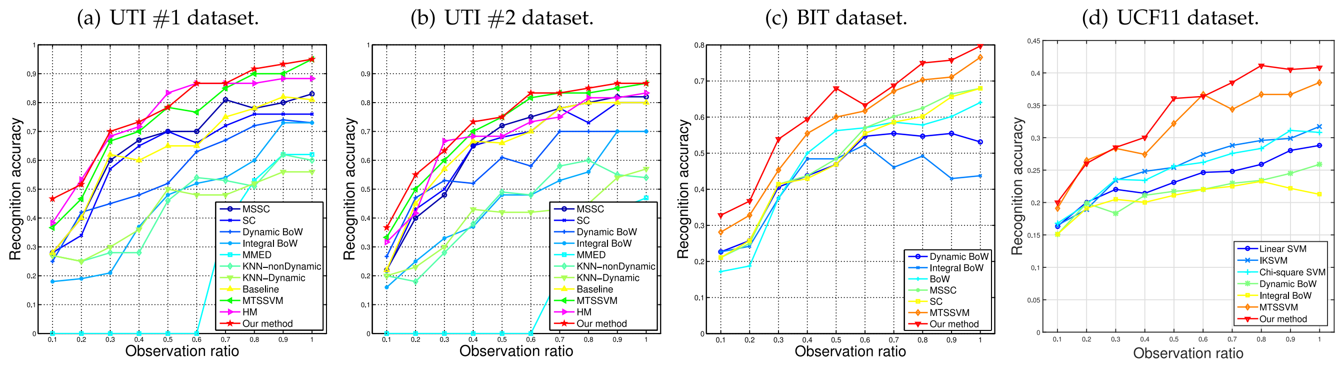


Fig. 6. Prediction results on the UTI $\#1$, UTI $\#2$, BIT, and UCF11 dataset. This figure is best viewed in color.

TABLE 1
Prediction Results on the UT1 $\#1$ Dataset with Different Observation Ratios

Methods	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
Linear SVM	30.00%	50.00%	53.33%	53.33%	50.00%	66.67%	60.00%	71.67%	78.33%	81.67%
IKSVM	30.00%	50.00%	63.33%	65.00%	61.67%	73.33%	71.67%	75.00%	71.67%	71.67%
$\mathcal{X}^2$-SVM	30.00%	55.00%	63.33%	58.33%	58.33%	73.33%	73.33%	78.33%	75.00%	75.00%
Dynamic BoW [35]	25.00%	41.67%	45.00%	48.33%	51.67%	63.33%	66.67%	71.67%	75.00%	73.33%
MSSC et al. [36]	28.33%	40.00%	60.00%	66.67%	70.00%	70.00%	81.67%	78.33%	80.00%	83.33%
MTSSVM [6]	36.67%	46.67%	66.67%	70.00%	78.33%	76.67%	85.00%	90.00%	90.00%	95.00%
no-cons2	45.00%	45.00%	60.00%	66.67%	70.00%	70.00%	68.33%	70.00%	81.67%	81.67%
no-cons3	45.00%	50.00%	60.00%	63.33%	70.00%	66.67%	70.00%	71.67%	76.67%	73.33%
no-LPM	46.67%	43.33%	61.67%	61.67%	56.67%	60.00%	66.67%	61.67%	63.33%	66.67%
no-GPM	45.00%	40.00%	60.00%	63.33%	70.00%	71.67%	66.67%	70.00%	70.00%	76.67%
Our method	46.67%	51.67%	70.00%	73.33%	78.33%	86.67%	86.67%	91.67%	93.33%	95.00%

the other comparison approaches in most cases. At 0.3, 0.5 and 1 observation ratios, MSSC achieves 48.33, 71.67, and 81.67 percent prediction accuracy, respectively; the prediction performance for SC are 50, 66.67, and 80 percent, and HM achieves 66.67, 68.33, and 83.33 percent, respectively. By contrast, our MMAPM achieves 66.67, 75 and 86.67 percent prediction results, respectively, which are consistently higher than MSSC, SC, and HM. MMAPM achieves 75 percent accuracy at with only the first 50 percent frames of testing videos are observed. This accuracy is even higher than the DBoW and IBoW with full observations. MMAPM achieves better prediction performance over MTSSVM in sevenout of 10 cases, and same results in three cases. This demonstrates that the composite kernels can help MMAPM learn better non-linear classification boundaries to differentiate partial observations of various classes. However, MTSSVM can only learn linear boundaries that may not be the optimal for predicting incomplete actions.

TABLE 2
Action Recognition Results on the UTI $\#1$ Dataset Compared with Existing Methods Given Full Observations

Methods	Accuracy
Ryoo & Aggarwal [14]	70.80%
Liu et al. [2]	73.33%
Yu et al. [10]	83.33%
Ryoo [35]	85.00%
Kong et al. [3]	91.67%
Raptis and Sigal [16]	93.33%
MTSSVM [6]	95.00%
Our method	95.00%

Note that our method achieves better prediction performance than [6] at early stages shown in Fig. 6a.

Component Evaluation. Our MMAPM consists of several key components, which includes Constraint (7) and (8), the local progress model (LPM in Eq. (4)), the global progress model (GPM in Eq. (3)), and the composite kernels in Eq (15). To verify their effectiveness in the full model, we remove each of these components from MMAPM, and obtain five variants, the no-cons2 model (remove the Constraint (7) from MMAPM), the no-cons3 model (remove the Constraint (8)), the no-LPM model (remove the LPM and Constraint (8)), the no-GPM model (remove the GPM and Constraint (7)), and the linear model, which is the MTSSVM proposed in [6]. The results of two recent action prediction methods [35], [36] are used for comparison. The full model is also compared with linear SVM, histogram intersection kernel SVM (IKSVM), and chi-square kernel SVM ($\mathcal{X}^2$-SVM)[4] [52]. Features $g(x_{(1,m)}, 1:m)$ used in GPM are the inputs for the three SVMs. The C parameters in the three SVMs are all set to 1.

Results in Table 1 indicate that our method achieves the best performance in comparison, which demonstrate that all the components in the full model are effective in action prediction. Compared with linear MTSSVM, our kernelized MMAPM achieves better performance in a majority of

4. The implementations of the three SVMs can be downloaded from http://www.cs.berkeley.edu/~smaji/projects/fiksvm/

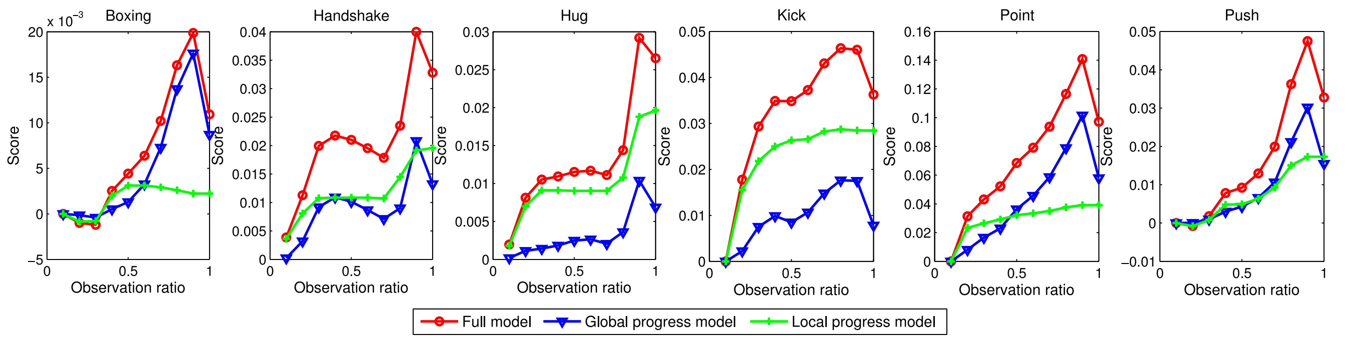


Fig. 7. Contributions of the global progress model and the local progress model to the prediction task.

observation ratios. This demonstrates the effectiveness of the composite kernels in MMAPM. Our method outperforms the no-cons2 and no-GPM methods, which shows the importance of capturing temporal evolution in the GPM. With GPM, the full model can utilize useful history information for prediction, and characterize the nature of sequentially arriving data. The full model achieves better performance than the no-cons3 and no-LPM. Thanks to the segment label consistency in Constraint (8)), the full model can extract discriminative information in segments, and utilize context information in segments.

We also compare with linear SVM and non-linear SVMs, and show results in Table 1. The better performance of MMAPM over SVMs indicate that MMAPM better captures complex structural information in the sequential data than SVMs. It also should be noted that $\mathcal{X}^2$-SVM and IKSVM achieve higher performance than the linear SVM in the first 8 observation ratios but perform worse in the rest two cases. This shows that the action features in the beginning may not be well separated by linear boundaries, and thus non-linear boundaries benefit the recognition task. However, action features become linear separable when 90 or 100 percent temporal sequences are observed. Using kernels for such data may cause overfitting problem and thus degrades prediction performance.

Contributions of GPM and LPM. To further demonstrate that both GPM and LPM are crucial for action prediction, we compare MMAPM with the model that only uses one of the two sources of information on the UTI #1 dataset. Fig. 7 shows the scores of the GPM and LPM ($\boldsymbol{\alpha}_m^{\mathrm{T}}\psi_1(x_{(1,m)}, y)$ for the GPM and $\sum_{l=1}^{M}\mathbf{1}(l\leqslant m)\cdot\boldsymbol{\beta}_l^{\mathrm{T}}\psi_2(x_{(1,m)}, y)$ for the LPM), and compare them to the scores of the full MMAPM model with respect to the observation ratio m. Results show that the LPM captures discriminative temporal segments for prediction. LPM characterizes temporal dynamics of segments and discriminatively learns to differentiate segments from different classes. In most cases, the score of LPM is monotonically increasing, which indicates a discriminative temporal segment is used for prediction. However, in some cases, segments from different classes are visually similar, and thus are difficult to discriminate. Therefore, in the middle of the "handshake" class and the "hug" class in Fig. 7 (observation ratio from 0.3 to 0.7), adding more segment observations does not increase LPM's contribution to MMAPM. Fig. 8 illustrates examples of visually similar segments in the two classes at $m = 6$. However, when such situations arise, GPM can provide the necessary appearance history information and therefore increases the prediction performance of MMAPM. The score of the full model drops when the final segments are observed. This is also caused by visual similarity as people are standing still in the last segment (see $m = 10$ in Fig. 8).

Sensitivity to the C Parameter. We further verify the sensitivity of MMAPM to the parameter C in Eq. (5). In this test, the C parameter is set to 0.01, 1, and 10. Results in Table 3 show that MMAPM is not sensitive to the C parameter when the observation ratio is low, i.e., only a small portion of videos are observed. The sensitivity increases when more segments are observed. In Table 3, the performance for the three parameter values at observation ratio 0.3 is the same. However, the difference increases to 8.33 percent when 80 percent of the videos are observed. The underlying reason is that, in the beginning of videos, the nonlinear boundaries learned from a small number of non-informative features may not be able to capture the complex motion variations of various action classes. Consequently, it may not be possible for MMAPM to generalize well using different C parameter values. As more expressive features are arriving, actions of different categories are being easily discriminated. However, an appropriate C parameter is still required for better generalization power as the structural features in MMAPM are very complex.

4.3 Results on BIT-Interaction Dataset

Action Prediction. We also compare MMAPM with the MSSC and SC [36], DBoW and IBoW [35], and MTSSVM [6] on the BIT-Interaction dataset. A BoW+SVM method is used as a baseline. Localization of people of interest is not performed for this dataset.

Results shown in Fig. 6c demonstrate that MMAPM outperforms MSSC and SC in all cases due to the effect of the GPM, which effectively captures temporal action evolution information. MMAPM also outperforms the DBoW and IBoW in all the cases. Our method achieves 59.38 percent recognition accuracy with only the first 40 percent frames of testing videos are observed, which is better than the DBoW and

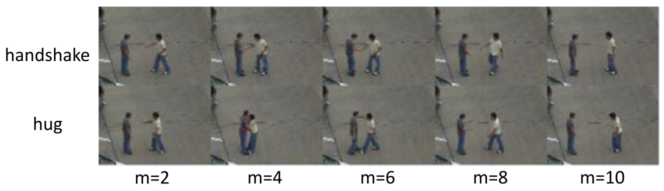


Fig. 8. Examples of visual similar segments ($m = 6, 8, 10$) in "handshake" and "hug".

TABLE 3
Accuracy of MMAPM on Videos of Observation Ratios 0.3, 0.5, and 0.8 with Different C Parameter Values

Observation ratio	$C = 0.01$	$C = 1$	$C = 10$
0.3	58.33%	58.33%	58.33%
0.5	61.67%	71.67%	71.67%
0.8	75.00%	76.67%	68.33%

IBoW at all observation ratios. Note that the performance of DBoW and IBoW do not increase much when the observation ratios are increased from 0.6 to 0.9. The IBoW performs even worse. This is due to the fact that some video segments from different classes are visually similar; especially the segments in the second half of the videos, where people return to their starting positions (see Fig. 9). However, since MMAPM models both the segments and the entire observation, its performance increases with the increase of observation ratio even if the newly introduced segments contain only a small amount of discriminative information. Significant improvements are expected over MTSSVM due to the use of kernels in MMAPM. MMAPM achieves 8.59, 7.97 and 4.69 percent better performance than MTSSVM with 30, 50 and 80 percent observations, respectively.

We also compare MMAPM with SVM methods with linear, histogram intersection, and chi-square kernel, respectively. Results in Table 4 indicate that our MMAPM outperforms all the other comparison methods. The performance difference between MMAPM and SVM methods demonstrates that MMAPM elegantly characterizes sequential data arrival in the prediction task. The superior performance of IKSVM and $\mathcal{X}^2$-SVM over linear SVM shows that non-linear kernels can learn better classification boundaries to categorize partial videos than the linear kernel.

Sensitivity to the C Parameter. We further investigate the sensitivity of MMAPM to the parameters C in Eq. (5). We set C to 0.05, 0.5, and 1, and test MMAPM with observation ratios 0.3, 0.5, and 0.8. Results in Table 5 indicate that MMAPM is not sensitive to the parameters when the observation ratio is low but the sensitivity increases when the observation ratio becomes large. In the beginning of a video, the small number of features available do not capture the variability of their class. Therefore, it does not help to use different parameters, because MMAPM cannot learn the appropriate class boundaries to separate all the testing data. As observation ratio increases, the features become more expressive. However, since structural features in MMAPM are very complex, appropriate parameters are required to capture the complexity of data.

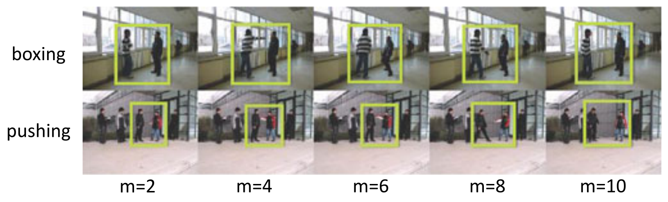


Fig. 9. Examples of visually similar segments in the "boxing" action (Top) and the "pushing" action (Bottom) with segment index $m \in \{2, 4, 6, 8, 10\}$. Bounding boxes indicate the interest regions of actions.

TABLE 5
Accuracy of MMAPM on Videos of Observation Ratios 0.3, 0.5, and 0.8 with Various C Parameter Values

Observation ratio	$C = 0.05$	$C = 0.5$	$C = 1$
0.3	43.75%	54.69%	42.97%
0.5	52.13%	65.63%	65.63%
0.8	56.25%	63.28%	73.44%

Component Evaluation. We also evaluate the importance of each component in MMAPM, including the Constraint (7), the Constraint (8), the local progress model (LPM in Eq. (4)), the global progress model (GPM in Eq. (3)), and the composite kernels in Eq. (15). We remove each of these components from MMAPM, and obtain five variant models, the no-cons2 model (remove the Constraint (7) from MMAPM), the no-cons3 model (remove the Constraint (8)), the no-LPM model (remove the LPM and Constraint (8)), the no-GPM model (remove the GPM and Constraint (7)), and the linear MTSSVM [6]. We compare MMAPM with these variants with parameter C of 0.08 and 10, and show prediction results in Fig. 10. The performance of the no-cons3 model and the no-LPM model are worse compared with the full method in all cases. This is due to the lack of the segment label consistency in the two models. The label consistency can help use the discriminative information in segments and also implicitly model context information. In the ending part of videos in the BIT dataset, since most of observations are visually similar (people return back to their normal position), the label consistency is of great importance for discriminating classes. However, due to the lack of the label consistency in the no-cons3 model and the no-LPM model, they cannot capture useful information for differentiating action classes. The superior performance of MMAPM over MTSSVM suggests the effectiveness of the composite kernels in the prediction task.

Progress Level Inference. In previous experiments, we assume that the progress levels of testing videos are known to all the methods. However, in most practical scenarios, progress levels are unavailable, which hinders the applications

TABLE 4
Prediction Results on the BIT Dataset with Different Observation Ratios

Methods	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
Linear SVM (BoW)	17.19%	18.75%	37.50%	50.00%	56.25%	57.03%	58.59%	57.81%	60.16%	64.06%
IKSVM	13.28%	25.00%	43.75%	57.03%	61.72%	68.75%	71.88%	67.19%	69.53%	68.75%
$\mathcal{X}^2$-SVM	14.84%	25.78%	41.41%	57.03%	61.72%	67.97%	71.09%	67.19%	67.18%	67.18%
Dynamic BoW [35]	22.66%	25.78%	40.63%	43.75%	46.88%	54.69%	55.47%	54.69%	55.47%	53.13%
MSSC et al. [36]	21.09%	25.00%	41.41%	43.75%	48.44%	57.03%	60.16%	62.50%	66.40%	67.97%
MTSSVM [6]	28.13%	32.81%	45.31%	55.47%	60.00%	61.72%	67.19%	70.31%	71.09%	76.56%
Our method	32.81%	36.72%	53.90%	59.38%	67.97%	63.28%	68.75%	75.00%	75.78%	79.69%

TABLE 6
Accuracy of Progress Level Inference Using $\mathcal{X}^2$-SVM with the Global Features (GF), the Local Segment Features (LSF), and the Combination of the Two Features (GF+LSF)

Features	Overall	$m=1$	$m=2$	$m=3$	$m=4$	$m=5$	$m=6$	$m=7$	$m=8$	$m=9$	$m=10$
GF	50.70%	96.09%	55.47%	44.53%	42.19%	34.38%	43.75%	47.66%	47.66%	40.63%	54.69%
LSF*	41.80%	96.88%	44.53%	28.13%	27.34%	30.47%	42.97%	32.03%	28.91%	32.81%	53.91%
GF+LSF*	52.03%	96.09%	56.25%	45.31%	44.53%	41.41%	41.41%	43.75%	42.97%	41.41%	67.19%

**Note that LSF and GF+LSF approaches are impractical in this setting since they require segment boundaries to be known, which already implies the availability of progress levels.*

of MMAPM. This problem can be solved by treating the inference of the progress levels as a classification problem independent of the action prediction problem (method 1). A M-class classifier can be trained to infer the progress level of a testing video, where each class corresponds to one out of M progress levels. Consequently, in practical scenarios, a two-step procedure is performed for a testing video: 1) progress level inference using the trained M-class progress level classifier; 2) action prediction using the proposed MMAPM given the predicted progress level. The problem can also be addressed by solving the inference of the progress level and the prediction of action jointly, i.e., maximizing over both m and y in Eq. (1) (method 2). However, method 2 is slower than method 1 in both training and testing due to larger search space, and thus is not preferred in applications that require prompt reaction.

We adopt method 1, and train a 10-class $\mathcal{X}^2$-SVM for 10 progress levels. The performance of three types of features, the global feature (GF) $g(x_{(1,m)}, 1:m)$, the local segment feature (LSF) $g(x_{(1,m)}, m)$, and the combination of these two features (GF+LSF) $g(x_{(1,m)}, 1:m) + g(x_{(1,m)}, m)$, are compared. Note that LSF and GF+LSF are impractical since segment boundaries are assumed to be known in LSF and GF+LSF, which implies the availability of progress levels.

Results in Table 6 demonstrate that the progress level is predictable on BIT dataset. The GF method achieves 50.70 percent overall accuracy in progress level prediction, close to the best GF+LSF method. LSF method performs worse than the other two methods as it only considers current segment feature. The prediction results of all the three methods on level 1 ($m=1$) are extremely high since most of training data for $m=1$ are 0-vectors, meaning that there are no STIPs or dense trajectories detected in the first segments. These 0-vectors are particular patterns that generally occur in the first segments.

Performance with Predicted Progress Levels. We further verify the influence of the progress level on the performance of MMAPM. We use GF and $\mathcal{X}^2$-SVM to predict the progress levels, evaluate the performance of MMAPM using predicted progress levels, and compare it with the one with the ground truth progress levels. MMAPM is compared with linear MTSSVM, and linear SVM, IKSVM, and $\mathcal{X}^2$-SVM are used as baseline action prediction methods. Results in Table 7 indicates that the performance of MMAPM with predicted progress levels (MMAPM(p)) does not drop greatly compared with the one with ground truth progress levels. The most significant decrease is 4.68 percent at $m=1$. MMAPM (p) achieves even better accuracy in some cases, e.g., $m=3$ and $m=6$. This indicates that the proposed MMAPM is capable of recognizing incomplete action videos in practical scenarios when the ground truth progress levels are unavailable. SVM-based methods with predicted progress levels, linear SVM(p), IKSVM(p), and $\mathcal{X}^2$-SVM(p), achieve better results in most cases compared with the corresponding methods with the ground truth progress labels. This is possibly because those methods do not use segment features, which are sensitive to the progress levels. However, if inaccurate progress levels are given to MMAPM(p) or MTSSVM (p), then the local progress model of MMAPM(p) or MTSSVM(p) could use the wrong model parameters for the segment features, and hence degrades the performance.

4.4 Results on UCF11 Dataset

MMAPM is also evaluated on the challenging UCF11 dataset [51]. We compare with Dynamic BoW (DBoW) and Integral BoW (IBoW) [35], and MTSSVM [6]. Linear SVM,

TABLE 7
Prediction Results on the BIT Dataset with Different Progress Levels

Methods	$m=1$	$m=2$	$m=3$	$m=4$	$m=5$	$m=6$	$m=7$	$m=8$	$m=9$	$m=10$
Linear SVM	17.19%	18.75%	37.50%	50.00%	56.25%	57.03%	58.59%	57.81%	60.16%	64.06%
Linear SVM(p)	13.28%	24.22%	40.63%	47.66%	56.25%	64.84%	67.19%	67.19%	67.19%	67.97%
IKSVM	13.28%	25.00%	43.75%	57.03%	61.72%	68.75%	71.88%	67.19%	69.53%	68.75%
IKSVM(p)	14.06%	22.66%	46.88%	55.47%	62.50%	74.22%	72.66%	67.97%	67.97%	68.75%
$\mathcal{X}^2$-SVM	14.84%	25.78%	41.41%	57.03%	61.72%	67.97%	71.09%	67.19%	67.19%	67.19%
$\mathcal{X}^2$-SVM(p)	14.06%	25.00%	46.88%	56.25%	62.50%	74.22%	73.44%	69.53%	65.63%	65.63%
MTSSVM	28.13%	32.81%	45.31%	55.47%	60.00%	61.72%	67.19%	70.31%	71.09%	76.56%
MTSSVM(p)	26.56%	31.25%	42.97%	55.47%	62.50%	62.50%	66.41%	69.53%	68.75%	74.22%
MMAPM	32.81%	36.72%	53.90%	59.38%	67.97%	63.28%	68.75%	75.00%	75.78%	79.69%
MMAPM(p)	28.13%	32.81%	57.03%	58.59%	67.97%	66.41%	68.75%	72.66%	73.44%	79.69%

Methods followed by "(p)" (e.g., MMAPM(p)) are the ones that are given predicted progress levels in testing computed by the global feature and $\mathcal{X}^2$-SVM. All the other methods are given the ground truth progress levels in testing.

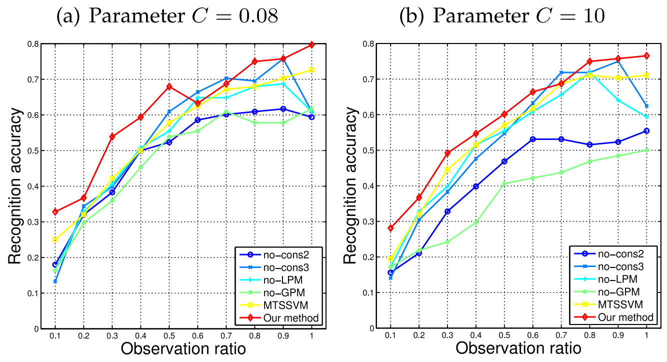


Fig. 10. Prediction results of each component in the full MMAPM with C parameter values equal to 0.08 and 10.

IKSVM and $\mathcal{X}^2$-SVM are used as baseline methods. The C parameters in these SVM methods and MTSSVM are tuned using the cross-validation scheme. The C parameter in MMAPM is set using the cross-validation scheme. Localization of people of interest is not used for this dataset and progress levels are known to all methods in testing.

As shown in Fig. 6d, our method and MTSSVM significantly outperform other comparison methods. Our method achieves 36.06 percent recognition accuracy when only 50 percent frames of testing videos are observed, which is even higher than DBoW, IBoW, linear SVM, IKSVM, and $\mathcal{X}^2$-SVM methods with full observations. MMAPM outperforms linear SVM, IKSVM, and $\mathcal{X}^2$-SVM as it captures the nature of sequential data arrival in the prediction task. DBoW and IBoW achieve relatively worse performance as the action models in these two methods are simply computed by averaging features, which are not able to capture large motion and pose variations in the UCF11 dataset. Our MMAPM achieves better results than MTSSVM in eight out of 10 cases due to the use of the composite kernels. These kernels allow us to better capture complex nonlinear class boundaries than linear MTSSVM. The performance improvements of kernelized SVMs, IKSVM and $\mathcal{X}^2$-SVM, over linear SVM also suggest the benefit of using kernels in action prediction task. It should be noted that the prediction results in Fig. 6d cannot be directly compared with action recognition results in [51] as we use an updated dataset of [51] here.

5 Conclusion

We have proposed the max-margin action prediction machine for recognizing actions in incomplete videos. MMAPM captures the entire action evolution over time and also considers the temporal nature of a video. We formulate the action prediction task as a structured SVM learning problem, and incorporate composite kernels to capture nonlinear classification boundaries in the prediction task. The monotonic increasing scoring function and discriminability of segments are enforced in the learning formulation. Experiments on four datasets show that MMAPM outperforms state-of-the-art approaches.

Appendix A
The Proof for the Proposition in Section 3.3

Proposition. Denote by ξ_1^*, ξ_2^* and ξ_3^* the optimal solutions of the slack variables in Eq. (5-8) for a given classifier f, then $(\xi_1^* + \xi_2^* + \xi_3^*)$ is an upper bound on the empirical risk $R_{\text{emp}}(f) = \frac{1}{N}\sum_{i=1}^{N}\sum_{m=1}^{M}\left[\Delta(\bar{y}_i, y_i)u(\frac{m}{M})\right]$.

Proof. Consider the fact in Constraint (6) that $\mathbf{w}^{\mathrm{T}}\Phi(x_{i(1,m)}, y_i) \leqslant \mathbf{w}^{\mathrm{T}}\Phi(x_{i(1,m)}, \bar{y}_i)$, we have

$$\frac{\xi_1^*}{u(\frac{m}{M})} \geqslant \frac{M}{N}\sum_{i=1}^{N}\delta(\bar{y}_i, y_i), \forall m, \forall \bar{\mathbf{y}}. \tag{20}$$

For all the M segments in the i-th training video, we have

$$\sum_{m=1}^{M}\xi_1^* = M\xi_1^* \geqslant \sum_{m=1}^{M}\frac{M}{N}\sum_{i=1}^{N}\delta(\bar{y}_i, y_i)u\left(\frac{m}{M}\right). \tag{21}$$

Therefore,

$$\xi_1^* \geqslant \sum_{i=1}^{N}\sum_{m=1}^{M}\frac{1}{N}\delta(\bar{y}_i, y_i)u\left(\frac{m}{M}\right). \tag{22}$$

Similarly, we can also prove that

$$\begin{aligned} \xi_2^* &\geqslant \sum_{i=1}^{N}\sum_{m=1}^{M}\frac{1}{N}\delta(\bar{y}_i, y_i)u\left(\frac{m}{M}\right), \\ \xi_3^* &\geqslant \sum_{i=1}^{N}\sum_{m=1}^{M}\frac{1}{N}\delta(\bar{y}_i, y_i)u\left(\frac{m}{M}\right). \end{aligned} \tag{23}$$

By combining (22) and (23) together, the following inequation can be derived:

$$\xi_1^* + \xi_2^* + \xi_3^* \geqslant \frac{1}{N}\sum_{i=1}^{N}\sum_{m=1}^{M}\Delta(\bar{y}_i, y_i)u\left(\frac{m}{M}\right) = R_{\text{emp}}(f). \tag{24}$$

This indicates that the training loss $(\xi_1^* + \xi_2^* + \xi_3^*)$ we minimize in the paper is an upper bound on the empirical risk $R_{\text{emp}}(f)$. □

Acknowledgments

This research is supported in part by the NSF CNS award 1314484, ONR award N00014-12-1-1028, ONR Young Investigator Award N00014-14-1-0484, NPS award N00244-15-1-0041, and U.S. Army Research Office Young Investigator Award W911NF-14-1-0218.

References

[1] C. Schuldt, I. Laptev, and B. Caputo, “Recognizing human actions: A local SVM approach,” in *Proc. Int. Conf. Pattern Recog.*, 2004, pp. 32–36.
[2] J. Liu, B. Kuipers, and S. Savarese, “Recognizing human actions by attributes,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2011, pp. 3337–3344.
[3] Y. Kong, Y. Jia, and Y. Fu, “Interactive phrases: Semantic descriptions for human interaction recognition,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 36, no. 9, pp. 1775–1788, Sep. 2014.
[4] N. Shapovalova, A. Vahdat, K. Cannons, T. Lan, and G. Mori, “Similarity constrained latent support vector machine: An application to weakly supervised action classification,” in *Proc. 12th Eur. Conf. Comput. Vis.*, 2012, pp. 55–68.
[5] T. Joachims, T. Finley, and C.-N. Yu, “Cutting-plane training of structural SVMs,” *Mach. Learn.*, vol. 77, no. 1, pp. 27–59, 2009.
[6] Y. Kong, D. Kit, and Y. Fu, “A discriminative model with multiple temporal scales for action prediction,” in *Proc. 13th Eur. Conf. Comput. Vis.*, 2014, pp. 596–611.

[7] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan, "Object detection with discriminatively trained part-based models," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 32, no. 9, pp. 1627–1645, Sep. 2010.

[8] P. Dollar, V. Rabaud, G. Cottrell, and S. Belongie, "Behavior recognition via sparse spatio-temporal features," in *Proc. 2nd Joint IEEE Int. Workshop Visual Surveillance Perform. Eval. Track. Surveillance*, 2005, pp. 65–72.

[9] H. Wang, A. Kläser, C. Schmid, and C.-L. Liu, "Action recognition by dense trajectories," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2011, pp. 3169–3176.

[10] T.-H. Yu, T.-K. Kim, and R. Cipolla, "Real-time action recognition by spatiotemporal semantic and structural forests," in *Proc. British Mach. Vis. Conf.*, 2010, pp. 1–12.

[11] I. Laptev, "On space-time interest points," *Int. J. Comput. Vis.*, vol. 64, no. 2, pp. 107–123, Sep. 2005.

[12] B. Wu, C. Yuan, and W. Hu, "Human action recognition based on context-dependent graph kernels," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2014, pp. 2609–2616.

[13] X. Wu, D. Xu, L. Duan, and J. Luo, "Action recognition using context and appearance distribution features," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2011, pp. 489–496.

[14] M. Ryoo and J. Aggarwal, "Spatio-temporal relationship match: Video structure comparison for recognition of complex human activities," in *Proc. IEEE 12th Int. Conf. Comput. Vis.*, 2009, pp. 1593–1600.

[15] M. Raptis and S. Soatto, "Tracklet descriptors for action modeling and video analysis," in *Proc. 11th Eur. Conf. Comput. Vis.*, 2010, pp. 577–590.

[16] M. Raptis and L. Sigal, "Poselet key-framing: A model for human activity recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2013, pp. 2650–2657.

[17] A. Vahdat, B. Gao, M. Ranjbar, and G. Mori, "A discriminative key pose sequence model for recognizing human interactions," in *Proc. IEEE Int. Conf. Comput. Vis. Workshops*, 2011, pp. 1729–1736.

[18] Y. Yang and M. Shah, "Complex events detection using data-driven concepts," in *Proc. 12th Eur. Conf. Comput. Vis.*, 2012, pp. 722–735.

[19] Y. Tian, R. Sukthankar, and M. Shah, "Spatiotemporal deformable part models for action detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2013, pp. 2642–2649.

[20] J. C. Niebles, C.-W. Chen, and L. Fei-Fei, "Modeling temporal structure of decomposable motion segments for activity classification," in *Proc. 11th Eur. Conf. Comput. Vis.*, 2010, pp. 392–405.

[21] Z. Wang, J. Wang, J. Xiao, K.-H. Lin, and T. S. Huang, "Substructural and boundary modeling for continuous action recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2012, pp. 1330–1337.

[22] K. Tang, L. Fei-Fei, and D. Koller, "Learning latent temporal structure for complex event detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2012, pp. 1250–1257.

[23] Q. Shi, L. Cheng, L. Wang, and A. Smola, "Human action segmentation and recognition using discriminative semi-Markov models," *Int. J. Comput. Vis.*, vol. 93, no. 1, pp. 22–32, May 2011.

[24] A. Vahdat, K. C nnons, G. Mori, S. Oh, and I. Kim, "Compositional models for video event detection: A multiple kernel learning latent variable approach," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2013, pp. 1185–1192.

[25] H. Pirsiavash and D. Ramanan, "Parsing videos of actions with segmental grammars," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2014, pp. 612–619.

[26] S. Bhattacharya, M. M. Kalayeh, R. Sukthankar, and M. Shah, "Recognition of complex events: Exploiting temporal dynamics between underlying concepts," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2014, pp. 2243–2250.

[27] B. Yao and L. Fei-Fei, "Recognizing human-object interactions in still images by modeling the mutual context of objects and human poses," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 34, no. 9, pp. 1691–1703, Sep. 2012.

[28] B. Yao and L. Fei-Fei, "Action recognition with exemplar based 2.5d graph matching," in *Proc. 12th Eur. Conf. Comput. Vis.*, 2012, pp. 173–186.

[29] X. Wu and Y. Jia, "View-invariant action recognition using latent kernelized structural SVM," in *Proc. 12th Eur. Conf. Comput. Vis.*, 2012, pp. 411–424.

[30] Z. Cai and Y. Qiao, "Multi-view super vector for action recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2014, pp. 596–603.

[31] C. Desai, D. Ramanan, and C. Fowlkes, "Discriminative models for static human-object interactions," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recog. Workshops*, 2010, pp. 9–16.

[32] A. Gupta, A. Kembhavi, and L. Davis, "Observing human-object interactions: Using spatial and functional compatibility for recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 31, no. 10, pp. 1775–1789, Oct. 2009.

[33] A. Prest, C. Schmid, and V. Ferrari, "Weakly supervised learning of interactions between humans and objects," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 34, no. 3, pp. 601–614, Mar. 2012.

[34] A. Patron-Perez, M. Marszalek, I. Reid, and A. Zissermann, "Structured learning of human interaction in TV shows," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 34, no. 12, pp. 2441–2453, Dec. 2012.

[35] M. S. Ryoo, "Human activity prediction: Early recognition of ongoing activities from streaming videos," in *Proc. Int. Conf. Comput. Vis.*, 2011, pp. 1036–1043.

[36] Y. Cao, D. Barrett, A. Barbu, S. Narayanaswamy, H. Yu, A. Michaux, Y. Lin, S. Dickinson, J. Siskind, and S. Wang, "Recognizing human activities from partially observed videos," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2013, pp. 2658–2665.

[37] K. Li and Y. Fu, "Prediction of human activity by discovering temporal sequence patterns," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 36, no. 8, pp. 1644–1657, Aug. 2014.

[38] T. Lan, T.-C. Chen, and S. Savarese, "A hierarchical representation for future action prediction," in *Proc. Eur. Conf. Comput. Vis.*, 2014, pp. 689–704.

[39] M. Hoai and F. D. la Torre, "Max-margin early event detectors," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2012, pp. 2863–2870.

[40] K. M. Kitani, B. D. Ziebart, J. A. Bagnell, and M. Hebert, "Activity forecasting," in *Proc. 12th Eur. Conf. Comput. Vis.*, 2012, pp. 201–214.

[41] B. Letham, C. Rudin, and D. Madigan, "Sequential event prediction," *Mach. Learn.*, vol. 93, no. 2/3, pp. 357–380, Nov. 2013.

[42] C. Rudin, B. Letham, A. Salleb-Aouissi, E. Kogan, and D. Madigan, "Sequential event prediction with association rules," in *Proc. 24th Annu. Conf. Learn. Theory*, 2011, pp. 615–634.

[43] I. Tsochantaridis, T. Joachims, T. Hofmann, and Y. Altun, "Large margin methods for structured and interdependent output variables," *J. Mach. Learn. Res.*, vol. 6, pp. 1453–1484, 2005.

[44] W. Yang, A. V. Yang Wang, and G. Mori, "Kernel latent SVM for visual recognition," in *Proc. Adv. Neural Inf. Process. Syst.*, 2012, pp. 818–826.

[45] A. Vedaldi and A. Zisserman, "Efficient additive kernels via explicit feature maps," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 34, no. 3, pp. 480–492, Mar. 2012.

[46] A. Vedaldi, V. Gulshan, M. Varma, and A. Zisserman, "Multiple kernel for object recognition," in *Proc. 12th Int. Conf. Comput. Vis.*, 2009, pp. 606–613.

[47] C. Ionescu, L. Bo, and C. Sminchisescu, "Structural SVM for visual localization and continuous state estimation," in *Proc. IEEE 12th Int. Conf. Comput. Vis.*, 2009, pp. 1157–1164.

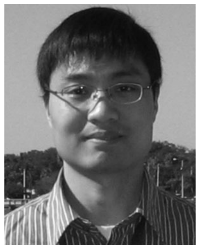
[48] L. Bertelli, T. Yu, D. Vu, and B. Gokturk, "Kernelized structural SVM learning for supervised object segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2011, pp. 2153–2160.

[49] M. Ryoo and J. Aggarwal, "Ut-interaction dataset, ICPR contest on semantic description of human activities," 2010. http://cvrc.ece.utexas.edu/SDHA2010/Human_Interaction.html

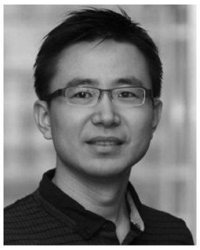
[50] Y. Kong, Y. Jia, and Y. Fu, "Learning human interaction by interactive phrases," in *Proc. 12th Eur. Conf. Comput. Vis.*, 2012, pp. 300–313.

[51] J. Liu, J. Luo, and M. Shah, "Recognizing realistic actions from videos "in the wild"," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2009, pp. 1996–2003.

[52] S. Maji, A. C. Berg, and J. Malik, "Classification using intersection kernel support vector machines is efficient," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2008, pp. 1–8.



Yu Kong received the BEng degree in automation from the Anhui University in 2006, and the PhD degree in computer science from the Beijing Institute of Technology, China, in 2012. He was a visiting student at the National Laboratory of Pattern Recognition (NLPR), Chinese Academy of Science from 2007 to 2009, and a visiting student at the Department of Computer Science and Engineering, State University of New York, Buffalo, in 2012. He is now a postdoctoral research associate in the Electrical and Computer Engineering, Northeastern University, Boston, MA. His research interests include computer vision, social media analytics, and machine learning. He is a member of the IEEE.



Yun Fu (S'07-M'08-SM'11) received the BEng degree in information engineering and the MEng degree in pattern recognition and intelligence systems from the Xi'an Jiaotong University, China, and the MS degree in statistics and the PhD degree in electrical and computer engineering from the University of Illinois at Urbana-Champaign, respectively. Prior to joining the Northeastern faculty, he was a tenure-track assistant professor in the Department of Computer Science and Engineering, State University of New York, Buffalo, during 2010-2012. His research interests are machine learning, computer vision, social media analytics, and big data mining. He has extensive publications in leading journals, books/book chapters, and international conferences/workshops. He serves as associate editor, chairs, PC member, and reviewer of many top journals and international conferences/workshops. He received five Young Investigator Awards (2016 IEEE CIS Outstanding Early Career Award, 2015 National Academy of Engineering US Frontiers of Engineering, 2014 ONR Young Investigator Award, 2014 ARO Young Investigator Award, 2014 INNS Young Investigator Award), five Best Paper Awards (SIAM SDM 2014, IEEE FG 2013, IEEE ICDM-LSVA 2011, IAPR ICFHR 2010, IEEE ICIP 2007), two Industrial Research Awards (2015 Adobe Faculty Research Award, 2010 Google Faculty Research Award), two Service Awards (2012 IEEE TCSVT Best Associate Editor, 2011 IEEE ICME Best Reviewer), etc. He is currently an associate editor of the *IEEE Transactions on Neural Networks and Leaning Systems (TNNLS)*, and *IEEE Transactions on Circuits and Systems for Video Technology (TCSVT).* He is a lifetime member of ACM, AAAI, SPIE, OSA, and the Institute of Mathematical Statistics, member of INNS and Beckman Graduate Fellow during 2007-2008. He is a senior member of the IEEE.